Notes on algebraic calculi of processes

Gérard Boudol

INRIA

CENTRE
SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.:(3) 954 90 20

Rapports de Recherche

N° 395

NOTES ON ALGEBRAIC CALCULI OF PROCESSES

Gérard BOUDOL

Avril 1985

NOTES ON ALGEBRAIC CALCULI OF PROCESSES

Gérard G. BOUDOL

INRIA - Sophia Antipolis
06560 Valbonne , France

Abstract

We gathered here some notes on Milner's calculi of processes. We interpret the terms of these calculi as transition systems. We intoduce a calculus called MEIJE built on a monoid of synchronized actions and illustrate some general semantic notions :

- we show the equivalence of this calculus with some others
- we give an implementation in a calculus restricted to purely atomic actions,
- we show the universality of MEIJE with respect to the notion of effective transition system and sketch its expressive power with regard to synchronization operators.

Finally the concept of subcalculus is illustrated through the description in our language of the class of rational parallel place machines.

Résumé

On présente quelques notes sur la notion de calcul algébrique de processus due à Milner. Ici les termes de ces calculs sont interpétés comme des systèmes de transitions. Ayant introduit un calcul appelé MEIJE, qui est construit sur un monoide d'actions synchronisées, on illustre quelques définitions générales concernant la sémantique :

- en montrant l'équivalence de ce calcul et de quelques autres,
- en donnant une implémentation dans un calcul à actions purement atomiques,
- en démontrant l'universalité de MEIJE par rapport à la notion de système de transition effectif et en esquissant son pouvoir d'expression quant aux opérateurs de synchronisation.

Enfin la notion de sous-calcul est illustrée par la description dans notre langage de la classe des machines à places rationnelles.

Contents

Foreword : This paper presents an overview, thus is not written according to the rules of mathematical rigour ; in particular it does not contain any proof.

1. Introduction

This paper sets some researches working on Milner's ideas about calculi of processes ([20] and especially [22]). We mainly address two questions regarding concurrent systems :

(1) what is a process ?

(2) how do systems compose one another and communicate ?

Concerning the first one we find a rather general concept at the cross-road of various approaches ([1,5,6,18,28]) : a process is a *labelled transition system* ([19]) or automaton (cf. [10]), in which states perform some discrete actions and become other states in doing so. This transition relation is denoted

$$p \xrightarrow{a} p'$$

However to be entitled to say that one has *parallel* systems, one must be able to take into account the global actions resulting from simultaneous activity of components. Moreover we shall assume that this activity is the only means to cooperate. Therefore actions themselves must carry out some kind of communication.

The first point is formalized by Milner in [21,22] by requiring that the set of actions is an abelian semigroup : to perform simultaneously two actions is to perform their product, obviously associative and commutative since it represents co-occurrence. It is perhaps more correct to understand composite actions as *non-interruptible* rather than *instantaneous* events (see § 6.2) and co-occurrence as the *synchronization* product, ensuring temporal atomicity. Composite actions are not just multisets : that some actions carry out communications means tha their co-occurrence creates something new. Here, since we deal only with pure synchronization, not value passing (as in [20], but see also [22]), communication will be handshaking : some actions have an inverse, and they set the abelian group of signal exchanges. Notations : $a.b$ for the product, a^- for the inverse.

Hence parallel processes are transition systems where transitions are labelled by actions belonging to some abelian monoid. To compose systems or

nets of processes, one usually put them into a synchronization structure or architecture. For instance the most basic way to combine two processes p and q is to build their *parallel composition* which we shall denote $(p \parallel q)$. More generally, any composition or synchronization mechanism appears to be a function on processes ; we call them *operators*, and given a family of primitive ones we construct compound systems by applying operators to constituents. This means that we describe processes by terms of a free algebra which sets up a *syntax* ; such an algebraic approach is now widely followed, even if semantical options differ (for instance see [6,8,15,22,25,26,30]).

On the semantical side we shall keep the informal hypothesis that "all happens through actions" : if the behaviour of a compound system depends on components, it only depends on their activity (no shared memory, for example). This is formalized by specifying the behavioural effect of an operator by means of structural rewriting rules, following Plotkin's style ([28,29]) of operational description. For instance in the "asynchronous" parallel composition we regard the arguments as independant. Thus the semantics of $(p \parallel q)$ is described by three rules according to the idea that either only one of the components runs, or both run synchronously :

if $p \xrightarrow{a} p'$ then $(p \parallel q) \xrightarrow{a} (p' \parallel q)$

if $q \xrightarrow{b} q'$ then $(p \parallel q) \xrightarrow{b} (p \parallel q')$

if $p \xrightarrow{a} p'$ and $q \xrightarrow{b} q'$ then $(p \parallel q) \xrightarrow{a.b} (p' \parallel q')$

Moreover a transition $(p \parallel q) \xrightarrow{c} r$ must be deduced by means of one of these specification rules. As another example, we shall also use a synchronization operator called *ticking* and denoted $a * p$ by which each action of a process p is linked to a given action a :

if $p \xrightarrow{b} p'$ then $a * p \xrightarrow{a.b} a * p'$

In most cases a will be an authorization signal sent by a synchronizer.

The first three technical sections of the paper set off the syntactical aspects of our MEIJE calculus : algebra of terms, built on a commutative monoid of action, and operational semantics. In such an algebraic setting, the above two questions become :

> what kind of transition systems and operators can we denote by expressions of our language ?

However we may only claim to describe the behavioural capabilities of transitions system or operators, even if they are operationally given. Indeed if one takes algebras of processes as an approach to programming languages (cf. [6,16,20]), this means that one wants to "program" for example synchronizers or synchronization mechanisms ignoring what are the internal states of such objects, provided they do their jobs. Therefore we have an informal postulate :

> nothing can be said about a system unless it results from observation of its actions.

This assumption may be interpreted in many ways (see for example [6,8,12,14]) ; here and in the work of Milner and others it is brought up through the notion of *bisimulation*, due to Park ([26], see also [5,22]). For we want to describe how systems operate, not how they can be observed. The idea of bisimulation on transition systems is the following : you have a (possibly infinite) set of colours ; then you paint each state of the system one colour. You get a bisimulation if your painting (partition) satisfies :

if there is a transition from, say, a blue state to a green state performing the action **a** then from *every* blue state your can perform **a** reaching a green state.

In this formulation "performing the action **a**" is actually relative to an *abstraction criterion* (or set of experiments, see [12]) : **a** may be an abstract action realized by some sequences of concrete actions, including perhaps invisible ones.

This leads to our semantical universe of processes regarded as quotients by bisimulations, which are still transition systems, on abstract actions. We also introduce in this universe the notions of *morphism* (various other attempts have been made in this direction, see [11,18,31,35] and references in these papers) and *simulation*. The latter is here the central semantical concept ; simulations are "concrete" state mappings which are morphisms on an "abstract" level.

Regarding the operators we interpret the above postulate by assuming that a semantics is not only a bisimulation but also a congruence, compatible with the given operators. This means for instance that specifications and verifications of a compound system are modularized, coming from specifications and verifications of the components. Thus an algebraic calculus of processes consists in :

- a syntax, which is a family of operators (relative to some set of actions) building a free algebra of terms,
- an operational semantics specifying the behavioural effect of the operators. Thus we get a transition system, the states of which are terms,
- a semantics given through a bisimulation compatible with the algebraic structure.

Particularizing simulations to this algebraic framework we get definitions for (syntax directed) translations (cf. [13]) and the notions of

- realizability of a transition system in a calculus
- definability of an operator
- subcalculus

which evolve from Milner's work [20,22].

This technical apparatus is presented in the central section (which may be partially omited at first reading). The rest of the paper is devoted to illustrate these general definitions.

In section 6 we show that our calculus could be presented by means of other primitive operators, with the same expressiveness. We also show that parallelism involving global actions may be simulated, up to an abstract view on action, by interleaving. After these basic examples (others may be found in [2]) we return to the motivating questions in section 7. There we show that every effective transition system is realizable in MEIJE and give de Simone 's result [33] concerning operators.

The universality of our calculus, as it is stated here, means that in some respects it is too strong, entailing the undecidability of some basic questions. Therefore we have to search for less powerful subcalculi. Moreover it is too abstract : we would like to have more concrete interpretations such as transition systems determined by some kind of *parallel machine* for instance. The last technical section is an attempt to progress in these directions. There we introduce the *rational parallel place machines* which generalize Petri nets. We conclude in discussing some semantical aspects of our propositions.

2. Actions

MEIJE is a synchronization calculus parametrized by a *commutative monoid* $\mathbb{M}$ of actions : it intends to provide some tools for organizing the concurrent behaviour of processes which perform actions in $\mathbb{M}$. These actions may themselves be communications. If for instance an equation $u=v$ is valid in $\mathbb{M}$ we may understand this equation as a *law of interaction* reflecting some communication structure (a similar idea is that of Winskel's synchronization algebra [35]). In this paper we will not go very far beyond this allusion with regard to action monoids.

We have to formalize the idea of *global* action of concurrent system. This activity results from the behaviour of components running together. Thus a first approximation is that a global action is a *set* of particulate actions. But distinct components may perform the same action. Therefore global ones are rather **multisets**, and even finite such since we shall assume that there are only finitely many active components in a system. Let A be some set of **atomic actions** ; here atomicity is temporal as well as spatial. These may be for instance names of programs written in some language. If we use $a,b,c...$ to range over A, a multiset on A is written

$$\{a,...,a,b,...,b,c,...,c...\}$$

But a more convenient notation is

$$a^n b^m c^k \ ... \text{ (in any order)}$$

where the positive integers $n,m,k...$ are the respective numbers of occurrences. And union is the sum of respective exponents, thereafter denote as a product :

$$(a^n b^m c^k).(a^{n'} b^{m'} c^{k'}) = a^{n+n'} b^{m+m'} c^{k+k'}$$

with unit 1.

From this point of view, multisets on A (or $\mathbb{N}$-sets, in Eilenberg's terminology, [10] chap. VI) are elements of the commutative monoid $\mathbb{N}<A>$. Technically $\mathbb{N}<A>$ is the set of mappings f from A to the additive monoid of positive integers $\mathbb{N}$ such that $f(a)=0$ for almost all $a \in A$. This monoid of multisets on A is also well-known to be the *free commutative monoid* generated by A. This is our first example of commutative monoid of actions.

Another basic example is that of free commutative group $fcg(S)$ generated by some set S of **signals**. Each of the elements of S, say s, is here a synchronization action, endowed with an inverse s^-. The interaction law $s.s^-=1$ (together with freeness) says that the primitive communication act is a handshake between two participants temporarily unavailable for other interactions. For instance (see [20,22]) we may have ports $p,q,r...$ such that for each value v of appropriate type there are two inverse actions p_v and p_v^- for sending and receiving the value v through the port p.

If we let for a while $s,p,q,...$ range over S then the actions of $fcg(S)$ are written

$$s^n p^m q^k \ ... \text{ (in any order)}$$

with $n,m,k,...$ integers. For it is well-known that $fcg(S)$ is isomorphic to $\mathbb{Z}<S>$, the set of mappings f from S to the additive group of integers $\mathbb{Z}$ such that $f(s)=0$ for almost all $s \in S$. This will be our **synchronisation group** : our calculi are parametrized by product monoids

$$\Gamma(\mathbb{M}) = \mathbb{M} \times \mathbb{Z}<\Lambda>$$

where

- $\mathbb{M}$ is any given commutative monoid
- $\Lambda = \{\lambda_n / n \in \mathbb{N}\}$ is a denumerable set of **signal identifiers** , or variables, disjoint from $\mathbb{M}$.

We shall use $\alpha,\beta,\gamma...$ and $u,v,w,...$ to range over Λ and $\Gamma(\mathbb{M})$ respectively. A pair $(u,v) \in \Gamma(\mathbb{M})$ is also written as a product $u.v$. Thus actions may be denoted

$u.\alpha^n\beta^m\gamma^k...$ with $u \in \mathbb{M}$ and $n,m,k...$ in $\mathbb{Z}$.

We say that $\alpha \in \Lambda$ **occurs** in $u \in \Gamma(\mathbb{M})$ if α is an irreducible factor of u, that is :

$$\forall B \subseteq \Lambda\ .\ u \in \mathbb{M}\times\mathbb{Z}<B> \Rightarrow \alpha \in B$$

For instance α occurs in $\alpha a b \alpha^{-2}$ but not in $\alpha\beta\alpha a \alpha^{-2}$.

Let us recall that a morphism over $\Gamma(\mathbb{M})$ is a mapping φ such that

$\varphi(1) = 1$ (where 1 is the unit of $\Gamma(\mathbb{M})$.)

$\varphi(u.v) = \varphi(u).\varphi(v)$

Thus if $u \in \Gamma(\mathbb{M})$ has an inverse u^-,$\varphi(u^-)$ is the inverse of $\varphi(u)$. In the **pure** MEIJE calculus over $\mathbb{M}$ we only use *pure synchronization morphisms*. These are morphisms which change nothing but a finite number of signal identifiers into packs of other ones. Thus they are denoted

$\varphi = <u_1/\alpha_1 ,..., u_k/\alpha_k>$ with $u_i \in \mathbb{Z}<\Lambda>$

For instance if $\varphi = <\alpha\beta/\alpha>$ and $u \in \mathbb{M}$ then

$\varphi(u\alpha^2) = u\alpha^2\beta^2$ and

$\varphi(u\alpha^-\beta) = u\alpha^-$

In the **applied** version of our calculus the basic monoid $\mathbb{M}$ is a product

$\mathbb{M} = \mathbb{N}<A> \times \mathbb{Z}<S>$

where A is a set of atomic actions, S a set of signals (and A ,S ,Λ are assumed to be pairwise disjoint). Here we shall use (applied) *synchronisation morphisms* φ given as

$<u_1/b_1 ,..., u_k/b_k>$

for some finite subset $B = \{b_1 ,..., b_k\}$ of $A \cup S \cup \Lambda$.

It will also be convienient to work with finite sets of actions, thus with the semiring $\mathbb{K}(\mathbb{M})$ of finite subsets of $\Gamma(\mathbb{M})$. We ambiguously use $u,v,w...$ to range over $\mathbb{K}(\mathbb{M})$ and denote as usual (cf. [10])

- 0 for the empty set of actions
- u for the singleton $\{u\}$ when $u \in \Gamma(\mathbb{M})$
- $u+v$ for the union of u and v
- $u.v$ for the product $\{x.y / x \in u \text{ and } y \in v\}$

We assume well understood the notation $u \in v$ for $u \in \Gamma(\mathbb{M})$ and $v \in \mathbb{K}(\mathbb{M})$. Here again we say that $\alpha \in \Gamma$ **occurs in** $u \subseteq \Gamma(\mathbb{M})$ if

$$\forall B \subseteq \Gamma \qquad u \subseteq \mathbb{M} \times \mathbb{Z}<B> \Rightarrow \alpha \in B$$

(α is an irreducible factor of at least one element of u).

3. Syntax

Syntacticaly a calculus of processes is a free algebra of terms, relative to a commutative monoid $\mathbb{M}$ of actions. In order to define (by recursion) infinite processes we assume given a countable set

$$X = \{x_n / n \in \mathbb{N}\}$$

of variables or *identifiers*. We let $x, y, z...$ to range over X. We have two versions of our calculus :

- a pure one, where we denote the set of terms by $\mathcal{M}_{\mathbb{M}}(X)$. In this case we only use pure synchronization morphisms : intuitively we only will be able to schedule processes in these calculi.
- an applied one, where the monoid $\mathbb{M}$ is a product $\mathbb{N}<A>\times\mathbb{Z}<S>$ and the set of terms is denoted $\mathcal{M}_{<A,S>}(X)$.

In this case we are allowed to use arbitrary synchronisation morphisms ; consequently we will be able to describe more discriminating synchronization mechanisms, depending on what actions are performed.

In both cases we use $p, q, r, ...$ to range over terms.

The syntax is the following :

(i) each identifier $x \in X$ is a term.

The constant $\mathbb{0}$ is a term

(ii) **action** : For each $u \in \mathbb{K}(\mathbb{M})$ if p is a term then $u : p$ is a term

(iii) **morphism** : if φ is a synchronization morphism and p is a term then φp is a term (if we only use pure morphisms, we are in a pure calculus, otherwise we are in an applied calculus, with the implicit hypothesis on the form of $\mathbb{M}$)

(iv) **recursive definitions** : if $x_{i_1}, ..., x_{i_k}$ are identifiers and $p, p_1, ..., p_k$ are terms then

$$(p \textbf{ where } x_{i_1} = p_1, ..., x_{i_k} = p_k) \textbf{is a term}$$

(v) **restriction** : for each signal identifier $\alpha \in \Lambda$ if p is a term then $p\backslash\alpha$ is a term.

These are, with slight lexical variations (but the same semantics, see below) among CCS's or SCCS's primitives. This is not the case of the following :

(vi) **asynchronous parallel composition** : if p and q are terms then $(p \parallel q)$ is a term

(vii) **ticking** : for each $u \in \mathbb{K}(\mathbb{M})$ if p is a term then $u * p$ is a term.

Let us have a words on static semantics (cf. [28]) : **bindings** in our language are recursive definitions and restriction. Therefore occurrences of variables (in X or Λ) may be free or bound and may accordingly be substituted or not. We shall not be very formalist on that matter since it is standard. We merely point out that :

- an occurrence of $\alpha \in \Lambda$ in a term comes through the action, morphism and ticking constructs. It is bound if it is under the scope of a restriction $\backslash\alpha$, that is in a subterm $q\backslash\alpha$.
- an occurrence of $x \in X$ in a term is bound if it is under the scope of a recursive definition, that is if it is an occurrence of an x_{i_j} in q or p_1... or p_k in a subterm $(q \textbf{ where } x_{i_1} = p_1, ..., x_{i_k} = p_k)$.

- syntactic equality of terms is, as usual, the *conversion* equivalence, that is identity up to the name of some bound variables in some subterms. For instance

  $(x \textbf{ where } x = a:x) = (y \textbf{ where } y = a:y)$

  $((\alpha + \beta) * \alpha^- : \mathbb{0})\backslash\alpha = ((\lambda + \beta) * \lambda^- : \mathbb{0})\backslash\lambda$

- in the course of substitution, we may have to convert bound variables in order to avoid captures of free variables. For instance, in an applied calculus let

  $p = (z \textbf{ where } z = ((<a.a/a>x \parallel <\alpha^-.b/b>y)\backslash\alpha \parallel \beta * z))$

  $q = \alpha * (a:z)$, $r = \alpha^- * (b:y)$

  and let σ be the substitution q/x , r/y. Then

  $$p[\sigma] = (z' \textbf{ where } z' = ((<\gamma a/a>q \parallel <\gamma^- b/b>r)\backslash\gamma \parallel \beta * z')).$$

We call **agent** any closed term (i.e. without any free variables) and **expression** any term without free signal identifiers. For instance

$$(\alpha * x \parallel \alpha^- * y)\backslash\alpha$$

is an expression, but not an agent.

We denote by $\mathcal{A}_{\mathbb{M}}$ and $\mathcal{E}_{\mathbb{M}}(X)$ respectively the sets of these terms (and $\mathcal{A}_{<A,S>}$, $\mathcal{E}_{<A,S>}(X)$ if we are in an applied calculus).

4. Operational semantics

In the operational interpretation each *agent* of the calculus is a state of a transition system, performing some actions in $\mathbb{M}$. Moreover these possible transitions only depend on the syntactic struture of the term. The word "operational" roughly means that we have a set of rules to get transitions $p \xrightarrow{u} q$ and a clause which says that valid transitions are only the ones we get after a finite game. Therefore it should be more correct to call operational semantics of an agent the set of proof trees of its transitions. In this setting a primitive operator or an expression without recursion determines a finite set of rules, thus a way to transform sets of proofs, and recursion comes through least fixed points.

A set of rules associated with an operator is its *semantic specification*, or operational description. In the following these rules will take the form :

$$\frac{p_1 \xrightarrow{u_1} q_1 , \ldots , p_k \xrightarrow{u_k} q_k}{p \xrightarrow{u} q} \text{ (with some additional conditions on actions)}$$

also written as :

$$p_1 \xrightarrow{u_1} q_1 , \ldots , p_k \xrightarrow{u_k} q_k \ (\ldots) \vdash p \xrightarrow{u} q$$

The consequence symbol $\vdash$ must be read, as usual, "if...then..." or more accurately "from...deduce...". The transition relation over MEIJE terms

$$\rightarrow \subseteq \mathcal{M}_{\mathbb{M}}(X) \times \Gamma(\mathbb{M}) \times \mathcal{M}_{\mathbb{M}}(X)$$

is the least one satisfying the following rule schemata :

(i) $R1 : u \in v \vdash v : p \xrightarrow{u} p$

This is the standard rule for action (or guarding) except that we allow finite sets of actions as guards. It means that $v : p$ is a process which may first perform an action in v and then behaves like p. The rule for morphisms is equally standard :

(ii) $R2 : p \xrightarrow{u} p' \vdash \varphi p \xrightarrow{\varphi(u)} \varphi(p')$

The most technical case is that of recursion. The behaviour of a term

$$(p \textbf{ where } x_{i_1} = p_1, \ldots, x_{i_k} = p_k)$$

is that of p in which the identifiers x_{i_j} stand for p_j. But this binding is recursive, thus in p_j itself an identifier x_{i_l} behaves like p_l. Therefore the rule is

(iii) let $r_j = (x_{i_j} \textbf{ where } x_{i_1} = p_1, \ldots, x_{i_k} = p_k)$

and

$$q_j = p_j[r_1/x_{i_1}, \ldots, r_k/x_{i_k}] \quad \text{for } 1 \leqq j \leqq k$$

Then

$$R3 : p[q_1/x_{i_1}, \ldots, q_k/x_{i_k}] \xrightarrow{u} p' \vdash (p \textbf{ where } x_{i_1} = p_1, \ldots, x_{i_k} = p_k) \xrightarrow{u} p'$$

The rule for restriction is also well-known ; the effect of the construct $\backslash\alpha$ is two-fold : from an external point of view it hides the signal α and from an internal point of view it constrains the exchange of the signal α

(iv) $R4 : p \xrightarrow{u} p'$ and α does not occur in $u \vdash p\backslash\alpha \xrightarrow{u} p'\backslash\alpha$

As we have already indicated there are three rules for the parallel composition which is thus a nondeterministic operator :

(v) $R5.1 : p \xrightarrow{u} p' \vdash (p \parallel q) \xrightarrow{u} (p' \parallel q)$

$R5.2 : p \xrightarrow{u} p', q \xrightarrow{v} q' \vdash (p \parallel q) \xrightarrow{uv} (p' \parallel q')$

$R5.3 : q \xrightarrow{v} q' \vdash (p \parallel q) \xrightarrow{v} (p \parallel q')$

Finally :

(vi) $R6 : p \xrightarrow{u} p'$ and $v \in w \vdash w * p \xrightarrow{uv} w * p'$

Examples 4

(1) No rule applies to $\mathbb{0}$. Thus this term does not perform any action. The same is true for $0 : p$ and $0 * p$.

(2) Let

$$p = (\alpha^- * (\alpha * a : \mathbb{0} \parallel \alpha * b : \mathbb{0}))\backslash\alpha$$

$$q = (\alpha^{-2} * (\alpha * a : \mathbb{0} \parallel \alpha * b : \mathbb{0}))\backslash\alpha$$

(with $\alpha \in \Lambda, a, b \in \mathbb{M}$)

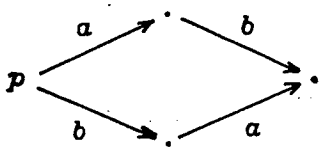
Then

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a:\mathbb{0} \xrightarrow{a} \mathbb{0}}R1}{\alpha * a:\mathbb{0} \xrightarrow{a.a} \alpha * \mathbb{0}}R6}{(\alpha * a:\mathbb{0} \parallel \alpha * b:\mathbb{0}) \xrightarrow{a.a} (\alpha * \mathbb{0} \parallel \alpha * b:\mathbb{0})}R5.1}{\alpha^- * (\alpha * a:\mathbb{0} \parallel \alpha * b:\mathbb{0}) \xrightarrow{a} \alpha^- * (\alpha * \mathbb{0} \parallel \alpha * b:\mathbb{0})}R6}{p \xrightarrow{a} \left[\alpha^- * (\alpha * \mathbb{0} \parallel \alpha * b:\mathbb{0})\right]\backslash\alpha}R4$$

Similarly one may prove, using R5.3, that

$$p \xrightarrow{b} (\alpha^- * (\alpha * a:\mathbb{0} \parallel \alpha * \mathbb{0}))\backslash\alpha$$

The reader can verify that the transition graph for p is



For q :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a:\mathbb{0} \xrightarrow{a} \mathbb{0}}R1}{\alpha * a:\mathbb{0} \xrightarrow{a} \alpha * \mathbb{0}}R6 \quad , \quad \dfrac{}{b:\mathbb{0} \xrightarrow{b} \mathbb{0}}R1}{(\alpha * a:\mathbb{0} \parallel \alpha * b:\mathbb{0}) \xrightarrow{a^2 a b} (\alpha * \mathbb{0} \parallel \alpha * \mathbb{0})}R5.2}{\alpha^{-2} * (\alpha * a:\mathbb{0} \parallel \alpha * b:\mathbb{0}) \xrightarrow{ab} \alpha^{-2} * (\alpha * \mathbb{0} \parallel \alpha * \mathbb{0})}R6}{q \xrightarrow{ab} (\alpha^{-2} * (\alpha * \mathbb{0} \parallel \alpha * \mathbb{0}))\backslash\alpha}R4$$

The reader can verify that this is the only possible transition from q.

(3) Let

$p = (x \textbf{ where } x = a:x)$.then

$$\vdash_{R_1} a:p \xrightarrow{a} p \vdash_{R_3} p \xrightarrow{a} p$$

Thus the transition graph of p is $p \circlearrowleft a$.

Similarly if we let $q = (x \textbf{ where } x = (a+b):x)$ one easily checks (using R1 and R3) that the transition graph of q is

$b \circlearrowleft q \circlearrowleft a$

More generally for each $u \in \mathbb{K}(\mathbb{M})$ we define the **clock on u** by

$$\boxed{h_u = (x \textbf{ where } x = u:x)}$$

Such processes will play a fundamental role. Their transition graphs are

$h_u \circlearrowleft v \in u$

i.e. they are processes which repeatedly perform some action in u (if any) and reconfigurate in themselves. As in [22] we denote :

$\mathbb{1} = h_1$

Another kind of "clock" is :

$p = (x \text{ where } x = 1 : (a * x))$

Here the number of a actions gives the current date, since we have :

$$p \xrightarrow{1} a * p \xrightarrow{a} a * a * p \xrightarrow{a^2} \ldots \xrightarrow{a^n} \ldots$$

(4) Let

$p = (x \text{ where } x = (a : \mathbb{0} \parallel x))$.

Then we have a proof tree T_1 :

$$\vdash_{R_1} a : \mathbb{0} \xrightarrow{a} \mathbb{0} \vdash_{R_{5.1}} (a : \mathbb{0} \parallel p) \xrightarrow{a} (\mathbb{0} \parallel p) \vdash_{R_3} p \xrightarrow{a} (\mathbb{0} \parallel p)$$

With T_1 we build a proof tree T_2

$$\cfrac{\cfrac{R1:\ \dfrac{}{a : \mathbb{0} \xrightarrow{a} \mathbb{0}}\ ,\ T_1}{R5.2:\ (a : \mathbb{0} \parallel p) \xrightarrow{a^2} (\mathbb{0} \parallel (\mathbb{0} \parallel p))}}{R3:\ p \xrightarrow{a^2} (\mathbb{0} \parallel (\mathbb{0} \parallel p))}$$

More generally if we have a proof tree T_n of a transition $p \xrightarrow{a^n} p_n$ we build, following the same scheme, a proof T_{n+1} of $p \xrightarrow{a^{n+1}} (\mathbb{0} \parallel p_n)$.

We shall see that semantically p denotes the process

$p \circlearrowright a^n \quad (n > 0)$

This suggests to generalize the definition of clocks to subsets U of $\mathbb{M}$ (not necessarily finite). For instance we denote, for $u \in \mathbb{K}(\mathbb{M})$:

$$\boxed{h_{u^*} = (x \text{ where } x = (u : \mathbb{0} \parallel x))}$$

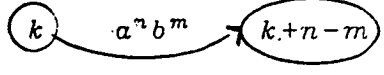
We will see later for what U the clock h_U is actually definable in MEIJE.

(5) Another basic example is that of a bag. Let us assume two actions a and b respectively meaning putting and removing a token. Then the behaviour of a bag p_k initially containing k tokens is as such :

One can simultaneously

- put (performing a) as many token as one wishes
- remove (performing b) at most k tokens.

In figure :



for $m \leqq k$, $n + m > 0$.

These processes are realized (see [3]) by

$$\begin{cases} p_0 = (x \text{ where } x = (a : b : \mathbb{0} \parallel x)) \\ p_{k+1} = (b : \mathbb{0} \parallel p_k) \end{cases}$$

(p_0 may be seen as an infinite parallel juxtaposition of elementary cells).

(6) We let the reader find the initial transitions of

$$p = (x \text{ where } x = (a:\mathbb{0} \parallel \langle ab/a, b^2/b\rangle x)).$$

Once provided this machinery we associate with each agent p a transition system, the states of which are terms reachable from p after a sequence of actions. The operational semantics of expressions is given by functions on these transition systems ; examples will be given in section 6 since this is only meaningful at an abstract level.

5. Semantics

5.1. Transition systems

We just have shown how operational semantics brings a structure of labelled transition system on the set of agents. Generally speaking a (labelled) **transition system** ([19]) (abbreviated t.s.) on a set **A** of actions is a triple

$\Theta = (\mathbb{Q}, \mathbf{A}, T)$ where

- $\mathbb{Q}$ is the set of *states*
- **A** is the set of *actions*
- T, the *transition relation*, is a subset of $\mathbb{Q} \times \mathbf{A}^* \times \mathbb{Q}$ where $\mathbf{A}^*$ is the set of finite sequences of actions.

We qualify Θ as **elementary** if $T \subseteq \mathbb{Q} \times \mathbf{A} \times \mathbb{Q}$

Notations : we abusively let $u, v, w, \ldots$ range over $\mathbf{A}^*$; the concatenation of v after u is $u:v$ and ε is the empty sequence. $p \xrightarrow[T]{u} q$ stands for $(p,u,q) \in T$. We extend the transitions to sequences by

$$(i) \quad p \overset{\varepsilon}{\underset{T}{\Rightarrow}} p$$

$$(ii) \quad \text{if } \exists q.\ p \xrightarrow[T]{u} q \ \&\ q \overset{v}{\underset{T}{\Rightarrow}} r \text{ then } p \overset{u:v}{\underset{T}{\Rightarrow}} r$$

and even to sets $u \subseteq \mathbf{A}^*$ of sequences by

$$(iii) \quad \text{if } \exists v \in u.\ p \overset{v}{\underset{T}{\Rightarrow}} q \text{ then } p \overset{u}{\underset{T}{\Rightarrow}} q$$

Remark 5.1 : usually a transition system is given provided with some *initial state*. But such systems $\Theta = (\mathbb{Q}, \mathbf{A}, T, q)$ are themselves states in a "universal" t.s. if we define

$$\Theta \xrightarrow{u} \Theta' \text{ iff } \Theta' = (\mathbb{Q}, \mathbf{A}, T, q') \ \&\ q \left(\xrightarrow[T]{u} q'\right).$$

Nevertheless all the following definitions and considerations are trivially extended to transition systems with initial state provided that in this case we assume the set $\mathbb{Q}$ **reduced** to the states reachable from the initial one through the extended relation $\underset{T}{\Rightarrow}$.

As we have pointed out, it is necessary to somehow abstract from such systems. For instance one may remark that in the examples above agents such as $\mathbb{0}$, $0:p$ and $0 * p$ or p and $(\mathbb{0} \parallel p)$ have the same behaviour. Therefore we want to

regard them as "equal". Equality holds through an abstract view on actions which we call an abstraction (or observation) criterion. Such a criterion on a set **A** of actions is a set of **abstract actions** (or observables, or experiments, see [12]) which themselves are non-empty sets of sequences of actions. The intended meaning is that all sequences of a single experiment $e \subseteq \mathbf{A}^*$ are held to carry out the same abstract action. Thus it is natural to assume that elements of a given criterion are disjoint ; stated differently : an abstraction criterion is deterministic. But it needs not be total : some sequences may be invisible or meaningless from a given point of view ; similarly a prefix of an observable sequence of actions may be unobservable (deadlock). Thus an **abstraction criterion** on a set **A** of actions is a *partial partition* over $\mathbf{A}^*$, i.e. a set $\mathbb{C} = \{e_i / i \in I\}$ of disjoint abstract actions $e_i \subseteq \mathbf{A}^*$. We also say that a sequence $u \in \mathbf{A}^*$ is *observable from* $\mathbb{C}$ if $\exists e \in \mathbb{C}. u \in e$.

The best known examples of this are criteria defining the strong congruence and the observation equivalence of CCS ([20], see also [22]). Both are special cases of criteria obtained from projections : let $B \subseteq \mathbf{A}$ be a set of *visible* actions ; then two sequences are observationally equivalent if their visible content is the same. Therefore observables are here classes in the equivalence relation

$$u \equiv_B v \iff \mu_B(u) = \mu_B(v)$$

where $\mu_B : \mathbf{A}^* \to B^*$ is the *projection* on B. This projection is the morphism of monoid determined by

$$\mu_B(a) = \text{if } a \in B \text{ then } a \text{ else } \varepsilon$$

When $B = \mathbf{A}$ ($\mu_{\mathbf{A}}$ is the identity) we get the **strong criterion** on **A** in which each sequence of actions is observable and distinguishable from any other one.

We intend to define a concept of congruence of a transition system in such a way that we would be able to define the quotient as a transition system on abstract actions. A **congruence** of a transition system $\Theta = (\mathbb{Q}, \mathbf{A}, T)$ (abbreviated t.s.-congruence) is a pair $\rho = (\mathbb{C}, R)$ where :

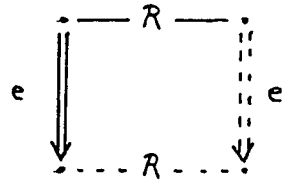
- $\mathbb{C}$ is an abstraction criterion on **A** and
- R is an equivalence relation on $\mathbb{Q}$ such that two R-equivalent states can not be distinguished by means of the observations from $\mathbb{C}$. Formally :

$$\forall q \in \mathbb{Q}\ \forall e \in \mathbb{C}\ \forall p \in \mathbb{Q}\ \forall q' \in \mathbb{Q}$$

$$\text{if } q \overset{e}{\underset{T}{\Rightarrow}} p \ \&\ (q, q') \in R$$

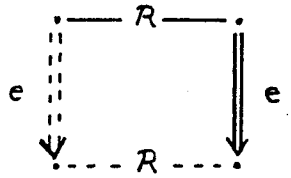
$$\text{then } \exists p' \in \mathbb{Q}.\ q' \overset{e}{\underset{T}{\Rightarrow}} p' \ \&\ (p, p') \in R$$

This commutation property is usually drawn



Brookes & Rounds [5] call such a relation R *invariant* ; it seems that these are *weak homomorphisms* of Ginzburg (see [26,31]). The terminological situation about "bisimulations" is rather confusing (compare for instance [5,22,26,31]).

Note that since we require R to be a symetric relation, the symetric schema also holds :



We call $\mathbb{C}$-congruence of transition system (or simply $\mathbb{C}$-congruence, and strong congruence when $\mathbb{C}$ is the strong criterion) an equivalence R on states such that $(\mathbb{C},R)$ is a t.s.-congruence.

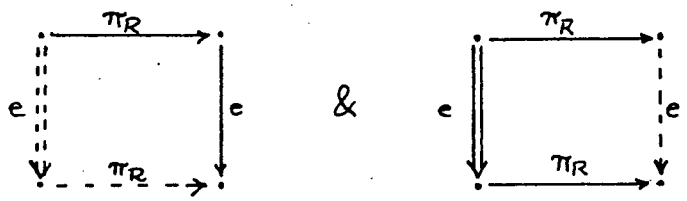
For a congruence $\rho = (\mathbb{C},R)$ on $\Theta = (\mathbb{Q},\mathbf{A},T)$ we define the **quotient** system Θ/ρ to be

$$\Theta/\rho = (\mathbb{Q}/R,\mathbb{C},T/\rho)$$

where (denoting by $\pi_R : \mathbb{Q} \to \mathbb{Q}/R$ the canonical surjection)

$$\pi_R(p) \xrightarrow[T/\rho]{e} \pi_R(q) \text{ if f } \exists r.\ p \overset{e}{\underset{T}{\Rightarrow}} r \ \&\ (r,q) \in R.$$

In diagrams :

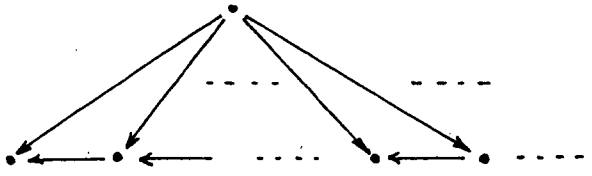


A quotient of a system Θ may also be called a *reduction* of this system.

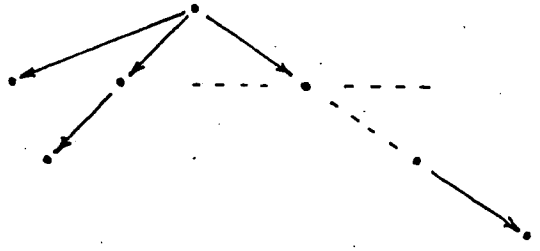
Remark 5.2 : if $\mathbb{C}$ is the strong criterion and Θ is elementary we shall regard Θ/ρ as an elementary transition system on $\mathbf{A}$.

Examples 5.1

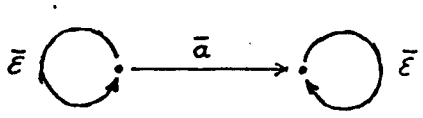
(1) With the strong criterion, the elementary transition system (where all the arrows are implicitely labelled by the same action)



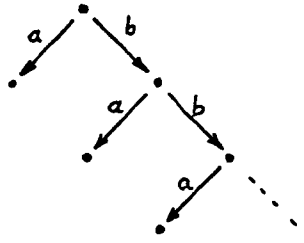
is a reduction of



(2) With the criterion on $A = \{a,b\}$ associated with the projection on $B = \{a\}$ in which observables are $\overline{\varepsilon} = \{b^n / n \in \mathbb{N}\}$, $\{b^{n_1}ab^{n_2}...b^{n_k}ab^{n_{k+1}} / n_i \in \mathbb{N}\} = \overline{a}^*$ the system



is a reduction of



(3) In MEIJE the equivalence relations on agents generated by

. $\{(p,(\mathbb{0} \parallel p))/p \in \mathcal{A}_M\}$

$\{(p \parallel q, q \parallel p)/p,q \in \mathcal{A}_M\}$

$\{(p \parallel (q \parallel r),(p \parallel q) \parallel r)/p,q,r \in \mathcal{A}_M\}$

are strong congruences.

An interesting fact (see [2,5,22]) is that an abstraction criterion $\mathbb{C}$ in itself determines a $\mathbb{C}$-t.s.-congruence in which the only means to distinguish states is observation of actions (for instance this constrasts with example (3) above where the "structure" of states is taken into account).

Lemma 5.1

For any transition system $\Theta = (\mathbb{Q},A,T)$ and for any observation criterion $\mathbb{C}$ on A there exists a coarsest $\mathbb{C}$-t.s.-congruence $\sim_{\mathbb{C}}$ on Θ i.e.

(i) $\sim_{\mathbb{C}}$ is a $\mathbb{C}$-congruence on Θ

(ii) For all $\mathbb{C}$-congruences R on Θ $(p,q) \in R \Rightarrow p \sim_{\mathbb{C}} q$

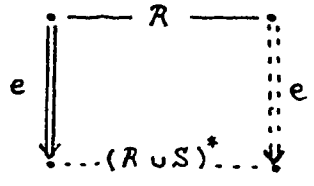
Notations : we shall use $\sim_{\mathbb{C}}$ uniformly disregarding the transition systems ; similarly the equivalence class of a state p in $\sim_{\mathbb{C}}$ is denoted $[]p[]_{\mathbb{C}}$ (respectively $[]p[]$ when $\mathbb{C}$ is the strong criterion). Following [5] we might use the name "$\mathbb{C}$-bisimulation" for $\sim_{\mathbb{C}}$; in order to avoid misunderstanding we prefer to call it $\mathbb{C}$-**equipollence.**

The above assertion may be stated as a very useful proof principle, which is nothing more than Park's induction :

Proposition 5.1 (the proof principle)

Let $\Theta = (\mathbb{Q},A,T)$ be a transition system and $\mathbb{C}$ an abstraction criterion on A. Let S be a $\mathbb{C}$-congruence on Θ and R a symetric relation over $\mathbb{Q}$.

If *for all* $e \in \mathbb{C}$



$$i.e. p \overset{*}{\underset{T}{\Rightarrow}} p' \;\&\; (p,q) \in R \Rightarrow \exists q'.\, q \overset{*}{\underset{T}{\Rightarrow}} q' \;\&\; (p',q') \in (R \cup S)^*$$

(where $(R \cup S)^*$ is the reflexive and transitive closure of $R \cup S$)

then $(R \cup S)^*$ is a $\mathbb{C}$-congruence

thus $(p,q) \in R \Rightarrow p \sim_{\mathbb{C}} q$ ▪

In this statement S is a set of already proved "equalities" which serve as hypotheses to prove $R \subseteq \sim_{\mathbb{C}}$.

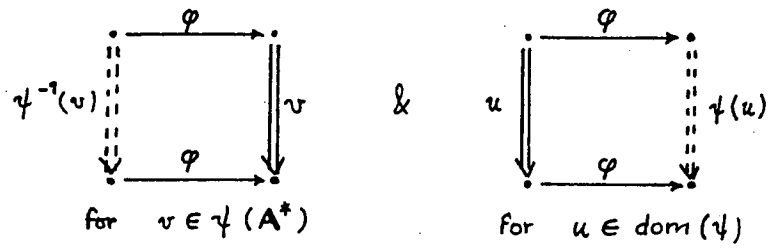
The notion of *morphism* is usually strongly related to that of congruence. Here the intended intuitive meaning is that a morphism from a system Θ to another one Θ' is a *representation* of Θ in Θ'. Therefore it is natural to assume :

- *soundness* : if the representation of a state q of Θ can carry out the representation of a sequence of actions u then q actually carry out a sequence v with the same representation.
- *completeness* : if a state q of Θ can carry out a representable sequence of actions u then its representation actually carries out the representation of u.

Let $\Theta = (\mathbb{Q},\mathbf{A},T)$ and $\Theta' = (\mathbb{Q}',\mathbb{B},T')$ be to transition system. A **morphism** from Θ to Θ' is a pair (ψ,φ) such that

- ψ is a *partial mapping* from $\mathbf{A}^*$ to $\mathbb{B}^*$ which tells which sequences of actions in Θ are represented by sequences in Θ', and how
- $\varphi : \mathbb{Q} \to \mathbb{Q}'$ tells how states of Θ are represented in Θ'

and this pair is subject to satisfy the properties :



(we omit the formal statements)▪

This notion of morphism is only an attempt among others called reduction, abstraction, contraction... see references in [11,18,31]. It is easy to check that it fulfils the usual requirements :

(1) for each $\Theta = (\mathbb{Q},\mathbf{A},T)$ the pair of identities on $\mathbb{Q}$ and $\mathbf{A}^*$ is a morphism. The composition $(\psi o\psi', \varphi o\varphi')$ of two morphisms is a morphism.

(2) Let $\rho = (\mathbb{C},R)$ be a congruence on Θ and

$\psi : \mathbf{A}^* \to \mathbb{C}$ (partial) given by

$\psi(u) = e$ if f $u \in e$

Then (ψ,π_R) is a morphism from Θ onto the quotient Θ/ρ.

(3) Let $\mathbb{Q}' \subseteq \mathbb{Q}$, $\mathbb{B} \subseteq \mathbf{A}$ and Θ' be the subsystem $(\mathbb{Q}',\mathbb{B},T')$ where $T' = T \cap (\mathbb{Q}' \times \mathbb{B}^* \times \mathbb{Q}')$.

Then the pair of canonical injections is a morphism.

(4) If (ψ,φ) is a morphism from Θ to Θ' and

$\mathbb{C} = \{\psi^{-1}(\psi(u)) \,/\, u \in dom(\psi)\}$

$(p,q) \in R \iff \varphi(p) = \varphi(q)$

then $\mathbb{C}$ is an abstraction criterion on $\mathbf{A}$ and R is a $\mathbb{C}$-congruence. Moreover $\Theta / (\mathbb{C},R)$ is isomorphic to the image subsystem $(\psi,\varphi)(\Theta)$ of Θ'.

In these notes we regard **semantics** of a system Θ as a quotient by a congruence ρ. Therefore a question raises : how can we "concretely" define a representation of Θ by another system Θ' in order to get a semantically sound representation ? Or technically stated (with the previous notations) :

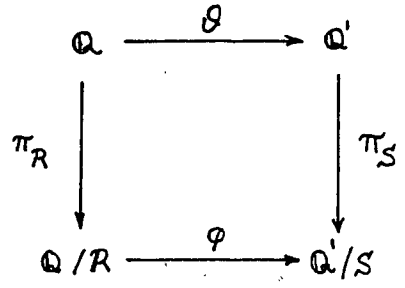
let

- $\rho' = (\mathbb{C}',S)$ *be a congruence on* Θ'
- ψ *a partial mapping from* $\mathbb{C}^*$ *to* $\mathbb{C}'$

We want to find "concrete" mappings $\vartheta : \mathbb{Q} \to \mathbb{Q}'$ such that there exists $\varphi : \mathbb{Q}/R \to \mathbb{Q}'/S$ satisfying

(i) (ψ,φ) is a morphism from Θ/ρ, the semantics of Θ, to Θ'/ρ'

(ii) the diagram



commutes ($\pi_S o \vartheta = \varphi o \pi_R$)

An immediate observation is that if such a φ exists then it is unique (notation : $\vartheta\hat{}$), thus "concretely determinded" by ϑ.

We call such a triple (ψ,ϑ,ρ') a **simulation** of (Θ,ρ) by or in Θ'.

Another observation is that a necessary and sufficient condition of exitence for φ satisfying (ii) is

$$(p,q) \in R \implies (\vartheta(p),\vartheta(q)) \in S$$

and if moreover the converse is true then φ is injective. (We have a "simulation criterion" analogous to proof principle for congruences, but it is rather technical ; and since we shall not give proofs explicitely using this criterion, we do not state it here).

The semantical notion of simulation is the central one of this paper, once particularized in two nested kinds :

- a simulation (ψ,ϑ,ρ') is a (correct) **implementation** of (Θ,ρ) in Θ' when ψ does not shade off the semantical criterion $\mathbb{C}$, i.e. is an injection from $\mathbb{C}$ to $\mathbb{C}'$, and similarly $\vartheta\hat{}$ is injective. Thus an implementation gives a concrete and semantically exact image of a system, up to an observation criterion on the target system, where actions of Θ/ρ are "implemented" as abstract actions.
- Let us assume that Θ and Θ' are t.s. on the same set $\mathbf{A}$ of actions and $\rho' = (\mathbb{C},S)$. Then ϑ is a $\mathbb{C}$-**realization** of (Θ,ρ) in (Θ',ρ') if $(id_{\mathbb{C}},\vartheta,\rho')$ is an implementation. Here we get a fully exact image of the semantics of Θ as a subsystem of (Θ',ρ') by means of the concrete state mapping ϑ. When $\mathbb{C}$ is the strong criterion we call ϑ a *strong realization*.

5. 2. Algebraic systems

In these notes we are mainly interested in **algebraic transition system** (abbreviated a.t.s.) $\Theta = (\mathbb{Q}, A, T)$ with respect to a family F of operators. This means that the set of states is the domain of an F-algebra. Thus for all $f \in F$ of arity k we have a mapping

$$f_\Theta : \mathbb{Q}^k \to \mathbb{Q}$$

(we omit the reference to Θ if it is understood)

N.B. : we assume here some familiarity with the standard concepts of algebra ; see for instance P.M. Cohn, "Universal Algebra" (Harper and Row, New York, 1965).

Let us take F as a set of *operator symbols* and X as a set of variables or parameters. Then we get syntactic objects which are the terms built on X by means of the constructors $f \in F$. These set up the domain $\mathcal{T}_F(X)$ of the well-known free F-algebra on X.

Notation : $\mathcal{T}_F$ for $\mathcal{T}_F(\phi)$, the set of ground terms. For instance this is the way in which MEIJE agents and expressions are built.

As usual terms $t \in \mathcal{T}_F(X)$ are interpreted on an algebraic system Θ as functions t^Θ of *evaluations* of the parameters in $\mathbb{Q}$. These are mappings $\nu : X \to \mathbb{Q}$ assigning to each variable a value. Let us recall the definition :

$$\begin{cases} x^\Theta(\nu) = \nu(x) & \text{for } x \in X \\ f(t_1,\ldots,t_n)^\Theta(\nu) = f_\Theta(t_1^\Theta(\nu),\ldots,t_n^\Theta(\nu)) \end{cases}$$

Obviously $t^\Theta(\nu)$ only depends on the value of variables occurring in t, thus t^Θ may also be regarded as a mapping from $\mathbb{Q}^k$ to $\mathbb{Q}$, if there are k distinct variables occurring in t.

Examples (in MEIJE) :

$t = (x \textbf{ where } x = (1{:}x \parallel y))$ will represent a function of one argument (named y) on processes, while

$t' = (\alpha^- * (\alpha * \alpha \parallel \alpha * y))\backslash\alpha$ will be a function of two arguments named x and y.

A value $t^\Theta(\nu)$ of such functions is called an *instance* in $\mathbb{Q}$ of the term t.

This allows to specify the transition relation of an algebraic system by a set of rules, as for the operational semantics of MEIJE (or CCS [20], SCCS [22] and so on). If we regard the operator f as a synchronization mechanism, each rule is one element of a non-deterministic strategy for f indicating

- which components of a system (composed by f) are allowed to proceed,
- what are the simultaneously permitted actions, and
- how the system recomposes a global transition (action and reconfigured state).

Therefore these rules take the following general format (cf. [33]) :

$$\mathbb{R} : \frac{x_{i_1} \xrightarrow{u_1} x'_{i_1},\ldots,x_{i_k} \xrightarrow{u_k} x'_{i_k}, (u_1,\ldots,u_k,u) \in U}{f(x_1,\ldots,x_n) \xrightarrow{u} t}$$

(a rule for f on A, also differently stated ...$\vdash$...) **where** :

– $x_1,\ldots,x_n$, $x'_{i_1},\ldots,x'_{i_k}$ *are distinct variables and* $\{x_{i_1},\ldots,x_{i_k}\} \subseteq \{x_1,\ldots,x_n\}$

We let for $1 \leqq j \leqq n$:

$$x'_j = \begin{cases} x'_{i_l} & \text{if } j = i_l \text{ for some } l \ (1 \leqq l \leqq k) \\ x_j & \text{otherwise} \end{cases}$$

- $U \subseteq \mathbf{A}^{k+1}$ *is a predicate on actions*

- t *is a term of* $\mathcal{T}_F(\{x'_1,\ldots,x'_n\})$ *in which each* x'_j *occurs at most once.*

(thus look-ahead in the future of the arguments is forbidden as well as duplications).

Examples 5.2

The generalized clocks h_U for $U \subseteq \mathbf{A}$ are specified by

$$u \in U \vdash h_U \xrightarrow{u} h_U$$

(considering h_U as a constant symbol)

We even generalize a little more to $U \subseteq \mathbf{A}^{k+1}$:

$$\frac{x_1 \xrightarrow{u_1} x'_1,\ldots,x_k \xrightarrow{u_k} x'_k,(u_1,\ldots,u_k,u) \in U}{h_U(x_1,\ldots,x_k) \xrightarrow{u} h_U(x'_1,\ldots,x'_k)}$$

(for instance ticking $u * x$ is such a synchronization mechanism)

We call **semantic specification** (of a set F of operators, on a set $\mathbf{A}$ of actions, abbreviated $(F,\mathbf{A})$-specification) a set Φ of such rules. We say rather informally that an *elementary* F-algebraic transition system $\mathcal{Q} = (\mathbb{Q},\mathbf{A},T)$ **satisfies** this specification iff

for any state $q = f_{\mathcal{Q}}(q_1,\ldots,q_n)$

$q \xrightarrow[T]{u} q'$ **iff** this transition is the conclusion drawn from an instance in $\mathbb{Q}$ of a Φ rule.

Remark : a "specification" here is thus required to be exact and complete. We shall not study the interesting notion of "satisfaction of a specification up to an abstraction criterion on actions".

Now when $\mathcal{Q} = (\mathbb{Q},\mathbf{A},T)$ is an F-algebraic transition system we obviously refine the concept of congruence : an **F-a.t.s.-congruence** (for short F-congruence or even congruence if no confusion is possible) on $\mathcal{Q}$ is a pair $\rho = (\mathbb{C},R)$ such that

- ρ is a t.s.-congruence
- R is compatible with the algebraic structure, i.e.
  for all $f \in F$ if $(p_1,q_1) \in R,\ldots,(p_n,q_n) \in R$ then
  $(f_{\mathcal{Q}}(p_1,\ldots,p_n),f_{\mathcal{Q}}(q_1,\ldots,q_n)) \in R$

In this case, where we still qualify R as $\mathbb{C}$-congruence or $(F,\mathbb{C})$-congruence, the quotient system Θ/ρ is also an F-algebraic system. We recall the standard definition :

$$f_{\Theta/\rho}(\pi_R(p_1),\ldots,\pi_R(p_n)) = \pi_R(f_\Theta(p_1,\ldots,p_n))$$

(we recall that $\pi_R(p)$ is the equivalence class of p in the relation R).

And we have

Lemma 5.2

Let Θ be an elementary F-algebraic system satisfying a semantic specification Φ of F.

(i) if R is a strong F-congruence on Θ then the quotient system satisfy the same specification

(ii) the strong equipollence $\sim$ is also an F-congruence (i.e. compatible with the algebraic structure).

Let us now introduce the notion of algebraic calculus of processes. A *simple algebraic calculus of process* is a structure $\mathcal{A} = (F,\mathbf{A},\Phi,\rho)$ where as before

- F is a family of operators
- $\mathbf{A}$ is a set of actions
- Φ is a semantic specification of F on $\mathbf{A}$
- $\rho = (\mathbb{C},R)$ is a *semantics*

Let us see what a semantics is : first we call *agents* of the calculus the ground terms belonging to $\mathcal{T}_F$. Then we define an operational transition system on agents

$$Op(\mathcal{A}) = (\mathcal{T}_F,\mathbf{A},T)$$

where T is the least transition relation satisfying the specification Φ. Then ρ is a semantics if it is an F-congruence on $Op(\mathcal{A})$.

The semantical universe of the calculus $\mathcal{A}$ is the quotient system

$$Sem(\mathcal{A}) = Op(\mathcal{A})/\rho = (\mathcal{T}_F/R,\mathbb{C},T/\rho)$$

in which the states (equivalence classes of agents) are the **processes** of the calculus. Equivalently process determined by an agent p might be seen as a subsystem of $Sem(\mathcal{A})$ with $\pi_R(p)$ as initial state (cf. remark 5.1). The transition system $Sem(\mathcal{A})$ is also an F-algebraic one. Moreover it satisfies the specification Φ if $\mathbb{C}$ is the strong criterion. Expressions of $\mathcal{T}_F(X)$ give **derived operators** on this system with which the semantics is still compatible.

However these calculi are too simple : there are no binding operators. When we have such operators the semantics of expression is given by means of substitutions. It seems fairly easy to extend a simple calculus to another one with recursive definitions of processes (see Prop. 4.6 in [22]). The situation is more delicate concerning operators such as restriction. A more careful analysis of the syntactical aspects of the semantic specifications is needed here. In particular one has to bring out the action parameters of an operator, occurring in the corresponding syntactic construct. This point is not yet clearly elucidated.

Therefore we shall keep a rather vague notion of *algebraic calculus of processes*. Such an object is a structure

$$\mathcal{A} = (F,\mathbf{A},T,\rho)$$

where

- F is a set of operators. We assume that some are bindings with respect to a set X of process identifiers and a set of action identifiers (disjoint from the set A of actions). Accordingly one defines the sets $\mathcal{A}_F$ of *agents* and $\mathcal{E}_F(X)$ of *expressions*. As in MEIJE agents are closed terms and expressions are terms in which action identifiers are bound.

  We also assume defined the operation of substitution and denote $t[p_1,...,p_n]$ the term $t[p_1/x_{i_1},...,p_n/x_{i_n}]$ if $x_{i_1},...,x_{i_n}$ are the free process variables occurring in t.
- $T \subseteq \mathcal{A}_F \times \mathbf{A} \times \mathcal{A}_F$ is the transition relation
- $\rho = (\mathbb{C},R)$ is as before a *semantics*, i.e.
  (i) R is a $\mathbb{C}$-congruence on $(\mathcal{A}_F,\mathbf{A},T)$
  (ii) R is compatible with the expressions : for all expression t, for all agents $p_1,...,p_n,q_1,...,q_n$

$$(p_1,q_1)\in R \ ,..., \ (p_n,q_n)\in R \ \Rightarrow \ (t[p_1,...,p_n],t[q_1,...,q_n])\in R$$

Then expressions t define *derived operators* $[\![t]\!]_{\mathcal{A}}$ on the quotient semantical system

$$Sem(\mathcal{A}) = (\mathcal{A}_F/R,\mathbb{C},T/\rho)$$

The extensional equivalence between expression is defined by

$$(t,t') \in R \quad \text{if f} \quad [\![t]\!]_{\mathcal{A}} = [\![t']\!]_{\mathcal{A}}$$

(when R is the $\mathbb{C}$-equipollence $\sim_{\mathbb{C}}$ we get the notation $\simeq_{\mathbb{C}}$)

Obviously we regard simple calculi as special cases of algebraic calculi. Concerning MEIJE we have :

Proposition 5.2

the strong equipollence ~ is a semantics of the MEIJE calculus.

In fact all the calculi considered in the following will be equipped with the **strong equipollence** as semantics. One may observe some apparently strange phenomena in MEIJE such as

$$x\backslash\alpha \simeq x$$

However this is not so surprising : since $t\backslash\alpha$ binds α in the "text" of the expression t , if α does not occur in t then obviously $t\backslash\alpha \simeq t$.

We conclude this section by particularizing the notions of implementation and realization to algebraic calculi of processes (an implementation of a calculus into another one might be called a compilation). Let $\mathcal{A} = (F,\mathbf{A},T,\rho)$ and $\mathcal{B} = (G,\mathbb{B},T',\rho')$ be two calculi of processes where $\rho = (\mathbb{C},R)$ and $\rho' = (\mathbb{C}',R')$ respectively. Then

(1) we first want to formalize the idea of **syntax-directed implementation**. Here each primitive construct of the source calculus is "implemented" as derived operator, ie. expression, of the target one. Let

- $\rho'' = (\mathbb{C}'',R'')$ be a t.s.-congruence on the semantical system $Sem(\mathcal{B})$
- $\psi : \mathbb{C} \to \mathbb{C}''$ an injective mapping
- $\vartheta : F \to \mathcal{T}_G(X)$ a mapping such that if the arity of $f \in F$ is n then $\vartheta(F)$ has exactly n free variables.

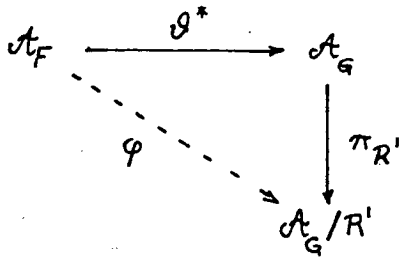
This mapping is algebraically extended as ϑ^* from $\mathcal{T}_F(X)$ to $\mathcal{T}_G(X)$ by

$$\begin{cases}\vartheta^*(x) = x & \text{for} \quad x \in X \\ \vartheta^*(F(t_1,\ldots,t_n)) = \vartheta(F)[\vartheta^*(t_1),\ldots,\vartheta^*(t_n)]\end{cases}$$

(note : we keep the same notation for its restriction to agents)

We suppose that ϑ^* preserves the notions of agent and expression. Finally let

- φ be given by the commutative diagram



Then (ψ,ϑ,ρ'') is a **translation** (or a **syntax-directed implementation**) of the calculus $\mathcal{A}$ in the calculus $\mathcal{B}$ if (ψ,φ,ρ'') is an implementation of the "concrete" (or syntactic) system $(\mathcal{A}_F,\mathbf{A},T)$ in the system $Sem(\mathcal{B})$.

(2) the calculus $\mathcal{A}$ is a *subcalculus* of $\mathcal{B}$ if each primitive construct of $\mathcal{A}$ is definable as a derived operator of $\mathcal{B}$. Let us assume that $\mathbf{A} = \mathbb{B}$ and $\mathbb{C} = \mathbb{C}'$. Then $\mathcal{A}$ is a **subcalculus** of $\mathcal{B}$ if there exists a mapping ϑ as in the point above such that ϑ^* (restricted to agents) is a $\mathbb{C}$-realization of $\mathcal{A}$ in $\mathcal{B}$.

The remainder of these notes is devoted to illustrate all these abstract semantical concepts.

6. Equivalent Formulations and Implementation

6.1. Subcalculi

Our first example of MEIJE-definable operator is Milner's **synchronous product** [21,22] $(p \times q)$ given by the specification (on any monoid $\mathbb{M}$).

$$p \overset{u}{\rightarrow} p',\ q \overset{v}{\rightarrow} q' \vdash (p \times q) \overset{uv}{\rightarrow} (p' \times q')$$

This is a kind of parallel composition in which components are not allowed to proceed independently. To define this operator in MEIJE we have to "tick" the two component -a priori independent- by the reception of a signal and to place them in a context which sends two signals synchronously. Thus we let

$$\boxed{x \times y \simeq (\alpha^2 * (\alpha^- * x \parallel \alpha^- * y)) \backslash \alpha}$$

(for some $\alpha \in \Lambda$)

In [2] we gave other formulae :

$$x \times y \simeq (\alpha * x \parallel \alpha^- * y) \backslash \alpha$$
$$\simeq ((\alpha^- * \tau \parallel \alpha^- * y) \parallel h_{\alpha^2}) \backslash \alpha$$

Remarks : this is the first step to show the equivalence of some calculi, in the sense that each will be a subcalculus of the other. Thus we could have formally defined a translation

$$\vartheta(\times) = (\alpha^2 * (\alpha^- * x_1 \parallel \alpha^- * x_2)) \backslash \alpha$$

and so on ; however we shall stay at a more intuitive level. A second remark is that if we would not take the restriction as a binding operator, then we would not be able to define uniformly operators by formulae. For instance if $\backslash\alpha$ does not bind α then $e[\alpha:\mathbb{0},\alpha:\mathbb{0}]$ where $e = (\alpha^2 * (\alpha^- * x \parallel \alpha^- * y))\backslash\alpha$ does not perform the synchronous product of $\alpha:\mathbb{0}$ by itself. On the contrary we avoid here sorting considerations (for processes as well as for the semantical equivalences of expressions).

Obviously ticking is strongly related to the synchronous product since

$$\boxed{u * x \simeq h_u \times x}$$

With this fundamental operator we can define another one which was taken as primitive in [2], **triggering** $(u=>p)$ of p by u , for $u \in \mathbb{K}(\mathbb{M})$. This synchronization has the effect of linking each initial transition of p to some action in u and then vanishes :

$$p \overset{v}{\rightarrow} p' , w \in u \vdash (u=>p) \overset{w.v}{\rightarrow} p'$$

The definition is :

$$\boxed{u => x \simeq (u:\mathbb{1}) \times x}$$

(let us recall that $\mathbb{1} = h_1$, cf. [22] and example 4.3).

This suggests to define for each $u \in \mathbb{K}(\mathbb{M})$ a process called the **trigger** (on u) by

$$\boxed{\tau_u = u:\mathbb{1}}$$

Then we derive an operator which is very often regarded as primitive, the well-known (binary) **sum** $(p+q)$:

$$p \overset{u}{\rightarrow} p' \vdash p+q \overset{u}{\rightarrow} p'$$
$$q \overset{v}{\rightarrow} q' \vdash p+q \overset{v}{\rightarrow} q'$$

In order to realize the sum, we have to trigger the two arguments on the reception of some signal and put them into a context which sends only one signal. The expression we gave in [2] was

$$x+y \simeq ((\alpha^-=>x \parallel \alpha^-=>y) \parallel \alpha:\mathbb{0})\backslash\alpha$$

Another one is :

$$\boxed{x+y \simeq (\alpha^-=>(\alpha^-=>x \parallel \alpha^-=>y))\backslash\alpha}$$

Thus we see that in our calculus non-deterministic choice results (with the presence of confluent non-determinism in the parallel composition) from unarbitrated communications.

Remark 6.1 : at this point we must mention the fact that our primitive operators $u:p$ and $u*p$ might have been restricted to $u \in \Gamma(\mathbb{M})$. If we are in a calculus applied to $\langle A,S \rangle$ they even may be restricted to

$u \in Y$ where $Y = A \cup S \cup S^- \cup \Lambda \cup \Lambda^-$

(with $\Lambda^- = \{\lambda^- / \lambda \in \Lambda\}$ and similarly for S^-).

For we have :

$$0:x \simeq 0*x \simeq \mathbb{0}\ ,\ 1*x \simeq x$$

$$1:x \simeq (\alpha^- :\mathbb{0} \parallel \alpha:x)\backslash\alpha$$

$$(u+v):x \simeq u:x + v:x$$

$$(u+v)*x \simeq h_{(u+v)}\times x$$

$$(u.v):x \simeq u=>(v:x)$$

$$(u.v)*x \simeq u*(v*x)$$

Yet another derived operator : the **interleaving** parallel composition $(p \mid q)$ in which the components are not allowed to proceed synchronously :

$$p \overset{u}{\rightarrow} p' \vdash (p \mid q) \overset{u}{\rightarrow} (p' \mid q)$$

$$q \overset{v}{\rightarrow} q' \vdash (p \mid q) \overset{v}{\rightarrow} (p \mid q')$$

The expression defining this operator in MEIJE is similar to that which defines the product but obviously we have here to send only one occurrence of the signal to the components :

$$\boxed{x \mid y \simeq (\alpha * (\alpha^- * x \parallel \alpha^- * y))\backslash\alpha}$$

This operator allows to express a **desynchronization** construct similar to that used in [21] by which the behaviour of a process is at any time delayed :

$$\vdash \nabla(p) \overset{1}{\rightarrow} \nabla(p)$$

$$p \overset{u}{\rightarrow} p' \vdash \nabla(p) \overset{u}{\rightarrow} \nabla(p')$$

It is trivially given by

$$\boxed{\nabla(x) \simeq (x \mid 1\!\!1)}$$

As we shall just see the situation is summarized by saying that we have (at least) three equivalent formulations of our MEIJE calculi. That is we have three calculi $\mathcal{M}$ (MEIJE), $\mathcal{S}$ (a variant of the early SCCS [21]) and $\mathcal{C}$, each of which being a subcalculus of the others :

	$\mathcal{M}$	$\mathcal{S}$	$\mathcal{C}$
action $u:p$ $(u \in \Gamma(\mathbb{M}))$	all primitive		
restriction $p\backslash\alpha$			
recursive definitions			
morphism φp	optional		
ticking $u*p$ $(u \in \Gamma(\mathbb{M}))$		derived	derived
parallel composition $(p \parallel q)$		derived	derived
desynchronization $\nabla(p)$	derived		derived
product $p \times q$	derived		
interleaving $p \mid q$	derived	derived	

(the blank spaces ought to be filled with "primitive")

Remarks : implicitely

- all these calculi are relative to a monoid $\Gamma(\mathbb{M})$
- the common semantics is the strong equipollence

We have seen how to derive the primitive operators of $\mathcal{S}$ and $\mathcal{C}$ in MEIJE, and some other equations :

$$u * x \simeq h_u \times x$$

$$u =\!> x \simeq \tau_u \times x \quad (\text{with } \tau_u = u : \mathbb{1})$$

$$\nabla(x) \simeq (x \mid \mathbb{1})$$

Thus for the other "inclusions" :

$$\boxed{(x \mid y) = (\nabla(h_{\alpha^-} \times x) \times \nabla(h_{\alpha^-} \times y) \times h_\alpha) \backslash \alpha}$$

This allows to define the sum in $\mathcal{S}$ (and in $\mathcal{C}$) :

$$x + y = (\alpha =\!> (\alpha^- =\!> x \mid \alpha^- =\!> y)) \backslash \alpha$$

Therefore we can define for example a clock

$$h_{\alpha+\alpha\beta+\beta} = (x \text{ where } x = \alpha : x + \alpha\beta : x + \beta : x)$$

Then

$$\boxed{((x \parallel y) = ((\alpha+\alpha\beta+\beta) * (\nabla(\alpha^- * x) \times \nabla(\beta^- * y))) \backslash \alpha, \beta}$$

The equivalence of $\mathcal{M}$ and $\mathcal{S}$ is that of two standpoints out of which

- the former sets an "asynchronous" parallel composition, where the components are priori independant and synchronized by some tools (ticking, restriction)
- The latter comprises a synchronous parallel composition and a construct to desynchronize (but also a synchronization mechanism : restriction).

The calculus $\mathcal{S}$ seems to be the "most primitive" one (exercice : write the exact formulation -without ticking and sum- of $\parallel$ in $\mathcal{S}$). The calculus $\mathcal{C}$ lies upon the distinction of the two fundamental aspects of parallel composition :

- mutual exclusion enforced in the interleaving operator, used as desynchronization
- temporal atomicity of global actions insured by the synchronous product

(see below, chapter 6.2)

We shall not distinguish these calculi, using the primitive of one or the other according to which is the more convenient.

Remark 6.2 : in SCCS [22] Milner uses some "infinitary" constructs : infinite sums or recursive definitions, and also more general restrictions parametrized by subsets of the action monoid :

$$p \overset{u}{\rightarrow} p', u \in B \vdash p \nvdash B \overset{u}{\rightarrow} p' \nvdash B$$

(this is a generalized clock, see example 5.2)

Here we want to have more "syntactical" (effective) constructs. We shall see what is lost in doing so. But pursing this spirit we might have define a calculus $\mathcal{M}'$ with

- action, parallel composition, recursive definitions and
- a unique synchronization construct called **interfacing**.

In such an operator $p \mid U$ (p *interfaced by* U) the given subset U of $\Gamma(\mathbb{M}) \times \Gamma(\mathbb{M})$ tells what actions are allowed to pass through the interface and how they are modified :

$$p \xrightarrow{u} p', (u,v) \in U \vdash p \mid U \xrightarrow{v} p' \mid U$$

This is a generalized clock (cf. example 5.2) which gathers restriction, morphism and ticking.

However such a construct is too abstract (for instance how does it bind signal identifiers ?).

Some other remarkable classes of processes are realized as subcalculi of MEIJE. A first example is that of finite processes, or finite automata but without terminal states. A **finite process** is given by a transition system with initial state $\Theta = (\mathbb{Q}, \mathbf{A}, T, q)$ such that $\mathbb{Q}$ and T are finite sets.. These are our ***finite*** objects in the semantical universe of transition systems. It might be rather clear that if $\mathbf{A} \subseteq \mathbb{M}$ then finite processes on $\mathbf{A}$ are strongly realized by agents of the subcalculus $\mathcal{F}_{\mathbf{M}}$ of MEIJE whose syntax is

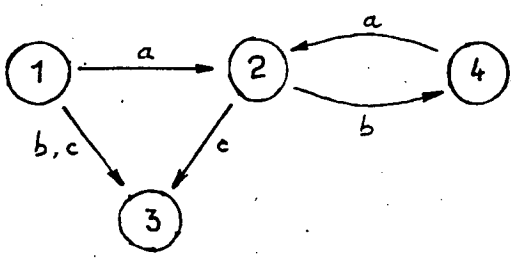
- the constant $\mathbb{0}$
- action $u:p$ for $u \in \mathbb{K}(\mathbb{M})$
- sum $p+q$, recursive definitions

Indeed Milner [23] has given a complete (with respect to the strong equipollence) axiomatization of this calculus ; as usual completeness lies upon the existence of *normal forms* for agents of the calculus. These are terms of the form

$$(x_l \text{ where } \ldots, x_i = \sum_{1 \le j \le k} u_i^j : x_j, \ldots)$$

in which it is easy to recognize a finite automaton : $x_1, \ldots, x_k$ represent states (with x_l initial) while the system of equations is nothing else than a linear grammar representing the transition table.

For example the equations of



are

$$x_1 = 0:x_1 + a:x_2 + (b+c):x_3 + 0:x_4$$

$$x_2 = 0:x_1 + 0:x_2 + c:x_3 + a:x_4$$

and so on.

With 1 as initial state, this process is also realized by the term

$$(b+c):\mathbb{0} + a:(x \text{ where } x = c:\mathbb{0} + b:y, y = a:x)$$

In a similar way one may describe the context-free grammars by a subcalculus $\mathcal{N}$ of MEIJE called in [2] that of *sequential non-deterministic processes*. Its constructors are

- the constant $\mathbb{0}$
- action $u:p$ for $u \in \mathbb{K}(\mathbb{M})$
- sum $p+q$ and sequential composition $p;q$
- recursive definitions

Let us explain what our **sequential composition** is (s.a. [6,20]). Assume we are in calculus applied to $\langle A, S \rangle$, where $s \in S$ is a distinguished **termination signal**. Then the specification of $p;q$ (implicitely parametrized by s) is

$$p \xrightarrow{u} p', s \text{ does not occur in } u \quad \vdash p;q \xrightarrow{u} p';q$$

$$p \xrightarrow{us^-}, s \text{ does not occur in } u, \; q \xrightarrow{v} q' \quad \vdash p;q \xrightarrow{u_1} q'$$

This operator is expressed in MEIJE (cf. [2]) by means of another one $p \setminus s$

$$x \setminus s \simeq (\alpha * \langle \beta s / s \rangle x \;\|\; (y \text{ where } y = \alpha^-:y + \alpha^-\beta:\mathbb{0}))\backslash \alpha,\beta$$

(x *until* S, which kills X as soon as it has signaled its termination)

$$x;y \simeq (\langle \alpha / s \rangle (x \setminus s) \;\|\; \alpha => y)\backslash \alpha$$

Remarks : here we see a use of applied morphisms in order to synchronize on specific actions. Our formulation, compared to that of CCS [20], shows the fruitfulness of Milner's idea of action monoid : here we do not have to look at this synchronization upto an observation criterion hiding internal communications. Moreover we do not have to modify the *text* of the synchronized agents : we simply put them into synchronizing expression.

Then for example the well-known grammar for the language of sequences of n a's followed by n b's ($n \geq 0$) is translated as

$$(x \text{ where } x = s^-:\mathbb{0} + (a:x);(b:s^-:\mathbb{0}))$$

And for the Dyck's language on $\{a,b\}$

$$(x \text{ where } x = s^-:\mathbb{0} + (a:x);(b:x) + (b:x);(a:x))$$

(see [2]).

We have given in [3] another example of a class of process realizable as a subcalculus of MEIJE, that of systems determined by *labelled Petri nets*. From an informal point of view, a calculus seems to correspond to a *language* where one frelly uses the primitive constructs to "program" processes or synchronization mechanisms wheras we have an image of the idea of *architecture* as expressions built with some "static" operators. For instance in MEIJE the static operators are restriction, morphism, ticking and parallel composition : they set an immutable structure upon their arguments. Let us call for a while *"synchronizing net expressions"* the expressions got from this syntax. Then any such term t with free process variables $x_1,\ldots,x_k$ each occurring once only may be shown equivalent to a normal form

$$(u_1 * \varphi_1 x_1 \;\|\; \ldots \;\|\; u_k * \varphi_k x_k \;\|\; syn)\backslash V$$

for some finite subset V of Λ, where $syn \in \mathcal{F}$ is a finite synchronizer

6.2. Implementation

It is often argued that parallelism is legitimately simulated by interleaving. Here we give a precise meaning to this assertion. It lies upon the duality of the notion of atomicity (= non-interruptibility) in the action monoid and the notion of abstraction criterion introducing atomicity at an abstract level.

A preliminary remark : in a MEIJE calculus (in any formulation) we may restrict the morphisms to be *alphabetic* , looking like relabellings of CCS, that is of the form $<... \ u/a \ ...>$ where $u \in \Lambda$. This was already pointed out by Milner in [22]. We leave its proof to the reader (hint : use the clocks h_{v^+}). We want to implement the $\mathcal{C}$ calculus applied to $<A,S>$ (with a termination signal $s \in S$) in which the action constructs $u:p$ are restricted to $u \in Y \cup \{1\}$ (see remark 6.1. We recall that $Y = A \cup S \cup S^- \cup \Lambda \cup \Lambda^-$).

The crucial observation is simply that the commutative monoid

$$\Gamma(\mathbb{M}) = \mathbb{N}<A> \times \mathbb{Z}<S \cup \Lambda>$$

is nothing else than a quotient of the free monoid Y^* of sequences on Y^*. Namely by the congruence given by the relations :

$u:v = v:u$ for all u,v

$u:u^- = \varepsilon$ (the empty sequence) for $u \in S \cup \Lambda$

Notation : $\pi(u) \subseteq Y^*$ is the equivalence class of u in this congruence.

In the target calculus $\mathcal{C}'$ only "particulate" actions (cf, [22]), i.e. in Y, are allowed. Then in our translation of $\mathcal{C}$ into $\mathcal{C}'$ a composite action u of $\Gamma(\mathbb{M})$ is represented by any sequence v such that $\pi(v) = u$. But this is not yet sufficient since we also have to distinguish sequences of composite actions. Therefore we need two particulate actions d and f, not in Y, indicating respectively the *beginning* and the *end* of a non-interruptible action in $\mathcal{C}$. For instance a sequence $d:a:a^-:b:a:f$ represents $a.b$. Thus an action $u \in \Gamma(\mathbb{M})$ is represented by any sequence of the set

$$\psi(u) = \{d:v:f \ / \ u = \pi(v)\}$$

Let

$$\$ = \{\{u\} / u \in \Gamma(\mathbb{M})\}$$

$$\mathbb{C} = \{\psi(u) / u \in \Gamma(\mathbb{M})\}$$

These are observation criteria respectively on $\Gamma(\mathbb{M})$ and $Y' = Y \cup \{d,f\}$. And $\psi:\$ \to \mathbb{C}$ is an injection.

Remarks : $\$$ determines the same equipollence $\sim$ as the strong criterion. Note also that some sequences of Y'^* are unobservable from $\mathbb{C}$.

Now in the target calculus $\mathcal{C}'$ the parallel composition acts as interleaving but with some rendez-vous mechanism in order to synchronize. Thus this calculus has some similarity with TCSP [6]. The syntax is :

(i) $\mathbb{0}$ is a term of $\mathcal{C}'$; each identifier $x \in X$ is a term of $\mathcal{C}'$

(ii) if $a \in Y'$ and $\neg$ is a term then $a:p$ is a term

(iii) if p is a term and φ an alphabetic synchronization morphism then φp is a term

(iv) if p and q are terms of $\mathcal{C}'$ then $p+q$ and $p;q$ are terms of $\mathcal{C}'$

(v) if p and q are terms of $\mathcal{C}'$ and B is a finite subset of Y' then $p \ |_B \ q$ is a term of $\mathcal{C}'$

(vi) recursive definitions, as usual.

Note : the composition operator $|_B$ binds the identifiers $\alpha\in\Lambda$ such that $\alpha\in B$ or $\alpha^-\in B$.

The semantics of the parallel composition operator $|_B$ is that the two components are interleaved except when they perform actions in B, on which there is a rendez-vous. This is similar to Milner's conjunction $\&_B$, see [22,2] and to de Simone ' s "produit de mixage" [32] :

$$p\overset{u}{\to}p',u\notin B \vdash (p\ |_B\ q)\overset{u}{\to}(p'\ |_B\ q)$$

$$p\overset{b}{\to}p',q\overset{b}{\to}q',b\in B \vdash (p\ |_B\ q')$$

$$q\overset{u}{\to}q',u\notin B \vdash (p\ |_B\ q)\overset{u}{\to}(p\ |_B\ q')$$

In order to define a syntax directed implementation of $\mathcal{C}$ in $\mathcal{C}'$ we obviously need some synchronizers :

- the first one ensures the correct scheduling of abstract actions, excluding overlapping :

$$sem = (x \text{ where } x = d : f : x)$$

This is just a simple semaphore, performing repeatedly the sequence $d:f$.

- the second one, used in the translation of restriction, controls the balance between occurrences of α and α^- ($\alpha\in\Lambda$) during the course of an abstract action :

$$syn_\alpha = (x \text{ where } x = (d:y);(f:x),$$
$$y = s^-:\mathbb{0} + (\alpha:y);(\alpha^-:y) + (\alpha^-:y);(\alpha:y))$$

(y performs sequences in a Dyck's language, in which α occurs as many times as α^-, and x is the iteration of y, see [2,6])

The translation ϑ is now fairly obvious :

(i) $\vartheta(\mathbb{0}) = \mathbb{0}$

(ii) $\vartheta(a:-) = d : a : f : x$ For $a\in Y$
and $\vartheta(1:-) = d : f : x$

(iii) $\vartheta(\varphi-) = \varphi x$

(iv) $\vartheta(- \text{ where } x_{i_1} = -,\ldots,x_{i_k} = -) = (z \text{ where } x_{i_1} = y_1,\ldots,x_{i_k} = y_k)$

(v) $\vartheta(-\backslash\alpha) = (x\,|_{B_\alpha} syn_\alpha)$ with $B_\alpha = \{d,f,\alpha,\alpha^-\}$

(vi) $\vartheta(-|-) = ((x\,|y)\,|_B sem)$ with $B = \{d,f\}$ (and $| = |_\phi$)

(vii) $\vartheta(-\times-) = (x\ |_B\ y)$

Our purpose is achieved since

proposition 6.1

the triple $(\psi,\vartheta,(\mathbb{C},\sim_{\mathbb{C}}))$ is a translation (syntax- directed implementation) of $\mathcal{C}$ in $\mathcal{C}'$.

The situation in this implementation is much simpler than, for instance, the more realistic problem of translating CSP in CCS (cf [13]).

7. Definability results

Until now we have seen only examples of what can be defined in MEIJE or any equivalent calculi, that is examples in the semantical universe of MEIJE processes and operators. But the two fundamental questions remain

- what is the class of MEIJE-realizable processes ?
- which operators can be defined by MEIJE expressions ?

To the first one we get, by means of de Simone ' s results [33,34], a rather complete answer. The second one is more difficult.

We study these problems in the particular case of processes on a free commutative monoid $\mathbb{N}<A>$ of actions with $A = \{a_1,...,a_m\}$.

It is well-known that such a monoid is isomorphic (by Parikh's mapping) to $\mathbb{N}^m$, the additive monoid of m-uples of positive integers. Thus we have a notion of recursively enumerable or computable subset of $\mathbb{N}<A>$:

$U \subseteq \mathbb{N}<a_1,...,a_m>$ **is computable iff**

$\{(n_1,...,n_m) \,/\, a^{n_1}{}_1 ... \, a^{n_m}{}_m \in U\}$ is a recursively enumerable subset of $\mathbb{N}^m$.

This definition is obviously extended to predicates, that is $U \subseteq \mathbb{N}<A>^k$, corresponding to subsets of $\mathbb{N}^{k.m}$.

In the sequel we let $\mathbb{M} = \mathbb{N}<a_1,...,a_m> \times \mathbb{Z}<S>$ (though the set S of signals does not play any rôle). The crucial result is :

Lemma 7.1 (de Simone [33,34])

Let $f : \mathbb{N}^k \to \mathbb{N}$ be a primitive recursive function, $a \in A$ and $\chi(a,f)$ the operator specified on $\Gamma(\mathbb{M})$ by the rule :

$$\left\{ \begin{array}{l} x_1 \xrightarrow{u_1 a^{n_1}} x'_1, \ldots, x_k \xrightarrow{u_k a^{n_k}} x'_k, \\ u_i \in \mathbb{N}<A-\{a\}> \times \mathbb{Z}<S \cup \Lambda>, \\ v = u_1 \ldots u_k \, a^{f(n_1,\ldots,n_k)} \end{array} \right\}$$

$$\vdash \chi(a,f)(x_1, \ldots, x_k) \xrightarrow{v} \chi(a,f)(x'_1,\ldots,x'_k)$$

Then $\chi(a,f)$ is strongly definable in the MEIJE applied calculus $\mathcal{M}_{<A,S>}$ ▪

The proof (see [33,34]) gives a construction of an expression e_f for f (we implicitely asssume a fixed) by induction on the scheme of definition of f :

(i) If f is the constant function $f(n_1,...,n_k) = 0$ then

$$e_f = <1/a> x_1 \times ... \times <1/a> x_k$$

(ii) If f is a projection $f(n_1,...,n_k) = n_i$ then

$$e_f = <1/a> x_1 \times ... \times x_i \times ... \times <1/a> x_k$$

(iii) If f is the successor function :

$$e_f = a * x_1$$

(iv) If f is a composition :

$$f(n_1,...,n_k) = g(g_1(n_1,...,n_k),...,g_l(n_1,...,n_k))$$

then

$$e_f = e_g[e_{g_1}/x_1,...,e_{g_l}/x_l]$$

(v) If f is defined by primitive recursion, for instance

$$\left\{ \begin{array}{l} f(0,n) = g(n) \\ f(l+1,n) = g'(f(l,n),l,n) \end{array} \right.$$

then

$$e_f = (<\alpha/a> x_1 \times <\beta/a> x_2 \times syn)\backslash\alpha,\beta$$

with

$$syn = (x \text{ where } x = e_g[t_1] + \alpha^- => (e_g[<\alpha\delta/\alpha,\lambda/\beta>x,t_2,t_3]\backslash\lambda,\delta))$$

where

$$t_1 = (y \text{ where } y = 1:y + \beta^- a => y)$$

$$t_2 = (y \text{ where } y = 1:y + \delta a => y)$$

and

$$t_3 = (u \text{ where } y = 1:y + \beta^-\lambda a => y) \quad \bullet$$

An easy corollary of this Lemma is :

Proposition 7.1

Let U be a computable subset of $\mathbb{N}<a_1,...,a_m>^{k+1}$ and H_U the operator specified on $\Gamma(\mathbb{M})$ by the rule

$$\left\{ \begin{array}{l} x_1 \overset{u_1 v_1}{\rightarrow} x'_1,...,x_k \overset{u_k v_k}{\rightarrow} x'_k, v_i \in \mathbb{Z}<S \cup \Lambda> \\ \text{and } u_i \in \mathbb{N}<A> \text{ (for } 1 \leq i \leq k), (u_1,...,u_k,u) \in U, \\ v = v_1 ... v_k u \end{array} \right\}$$

$$\vdash H_U(x_1,...,x_k) \overset{v}{\rightarrow} H_U(x'_1,...,x'_k)$$

Then H_U is strongly definable in $\mathcal{M}_{<A,S>}$ •

Remark 7.1 : in fact this is true in the restricted calculus $\mathcal{S}'_{<A,S>}$ which is $\mathcal{S}_{<A,S>}$ without the desynchronization operator ∇ but with sum as a primitive operator.

This result has quite a lot of consequences. For instance if $U \subseteq \mathbb{N}<A>$ it means that any **effective clock** h_U (with U computable) is strongly realizable in MEIJE. This contrasts with SCCS [22] in which one always may define :

$$h_U = fix\ x\ .\ \{x = \sum_{u \in U} u:x\}$$

even when U is not computable.

As we have seem clocks are related to ticking operators, thus we are able to generalize this construct :

$$\boxed{U * x \simeq h_U \times x}$$

Similarly every **effective trigger** τ_U is realizable :

$$\boxed{\tau_U \simeq (\alpha * h_U \ \|\ \alpha^-:1])\backslash\alpha}$$

and the associated triggering construct is

$$\boxed{U => x \simeq \tau_U \times x}$$

which allows to generalize the action operator :

$$\boxed{U:x \simeq U=>(1:x)}$$

When $U \subseteq \mathbb{N}<A>^2$ the above proposition indicates the definability of some interfacing operators. For instance if B is a computable subset of $\mathbb{N}<A>$ then the "restriction" operator $p \restriction B$ which is $p \mid U$ for

$$U = \{(u.v, v.u) / u \in B, v \in \mathbb{Z}<S \cup \Gamma>\}$$

is definable.

Yet another example : **effective pure scheduling**. A pure scheduling of a system of k processes is a context which tells at any discrete time $n \in \mathbb{N}$ what are the components allowed to proceed. Such a control is determined by a (recursively enumerable) subset V of $\{0,1\}^k \times \mathbb{N}$: the last component is the current date, the k first ones are boolean values (not allowed / allowed to proceed). We omit the obvious formal specification. Assuming that A contains an action α , the subset V is identified to a computable subset U of $\mathbb{N}<\alpha>^{k+1}$. Then the itended scheduling operator is defined by the expression

$$(a_1 * x_1 \parallel \ldots \parallel a_k * x_k \parallel synchro) \backslash a_1, \ldots, a_k$$

with

$$synchro = (<\alpha / \alpha> H_U[t_1, \ldots, t_k] \times t) \backslash \alpha$$

and

$$t_i = h_{(1 + a_i^- a)} \qquad (1 \leq i \leq k)$$

$$t = (x \text{ where } x = 1 : (\alpha^- * x)) \quad \text{(sceexample 4.3)}$$

Here we do not have morphisms on the arguments since we do not care about synchronisation of specific actions.

Obviously the strong expressive power of our calculus immediately entails undecidability results :

Proposition 7.2

The (equivalent) questions of whether or not, for agents $p \in \mathcal{A}_{<A,S>}$

(1) $p \sim \mathbb{0}$

(2) $\{u / \exists q. p \xrightarrow{u} q\} = \phi$

(3) $\{q / \exists u. p \xrightarrow{u} q\} = \phi$

are undecidable.

Indeed if B is a recursively enumerable but not recursive subset of $\mathbb{N}<A>$ and $u \in \mathbb{N}<A>$ we have

$$(u : \mathbb{0}) \restriction B \sim \mathbb{0} \iff u \notin B \quad \blacksquare$$

The main applications of proposition 7.1 are to answer the questions asked at the outset of this section. We define an **effective transition system** on $\mathbb{N}<A>$ (with $A = \{a_1, \ldots, a_m\}$) to be a system $\Theta = (\mathbb{Q}, \mathbb{N}<A>, T)$ such that

- $\mathbb{Q} = \{q_n / n \in \mathbb{N}\}$ is a denumerable set of states and
- the transition relation is effective with respect to the enumeration of $\mathbb{Q}$. That is

$$V = \{(n, n_1, \ldots, n_m, k) / (q_n, a^{n_1}{}_1 \ldots a^{n_m}{}_m, q_k) \in T\}$$

is a recursively enumerable subset of $\mathbb{N}^{m+2}$.

Theorem 7.1 (first definability theorem)

any effective transition system on $\mathbb{N}<A>$ (A finite) is strongly realizable in an applied MEIJE calculus.

The idea is the following : we introduce two new actions a and b (not in A) which will serve as "coloured sticks" to count the index of the starting state and target state of the transitions. Now let (with the previous notations)

$$U = \{a^n a^{n_1}_1 \ldots a^{n_m}_m b^k \;/\; (n,n_1,\ldots,n_m,k) \in V\}$$

By definition this is a computable subset of $\mathbb{N}<a,a_1,\ldots,a_m,b>$ if V is recursively enumerable. Thus if Θ is effective, we know by proposition 7.1 that the clock h_U is strongly realizable in MEIJE. Obviously we shall convert a and b to receiving of signals α and β, and it only remains to ensure the correct chaining of states. In order to do this we need a synchronizer t which satisfies :

$$\forall n \in \mathbb{N} \quad t \overset{\beta^n}{\rightarrow} (\alpha^n => t)$$

(in fact $\alpha^n => t$ is a "synchronous bag" containing initially n tokens, cf. example 4.5)

It is easily proved that we may let

$$t = (x \text{ where } x = (\beta : \alpha : \mathbb{1}) \times x + 1 : x)$$

Finally with each state q_n of Θ we associate

$$\vartheta(q_n) = (<\alpha^- / a, \beta^- / b> h_U \times (\alpha^n => t)) \backslash \alpha,\beta$$

and this is the intended strong realization ▪

Remark : this result holds in $\mathcal{J}'_{<A',S>}$ (with $A' = A \cup \{a,b\}$), see remark 7.1.

To conclude this section we quote de Simone ' s theorem about definability of operators. Let us call for a while **effective calculus of processes** any simple algebraic calculus $A = (F, \mathbb{M}, \Phi, \rho)$ such that

- F is a finite set of operator symbols
- $\mathbb{M} = \mathbb{N}<A>$ for some finite alphabet of actions A
- Φ is a finite semantic specification of F on $\mathbb{M}$ such that for each rule of Φ the predicate U on actions is computable (cf. § 31)
- ρ is the strong congruence

Theorem 7.2 (second definability theorem, de Simone [33])

any effective calculus of processes is a subcalculus of an applied MEIJE calculus.

To prove this de Simone builds a syntax-directed translation

$$\vartheta(f) = (\alpha_1 * \varphi_1 x_1 \parallel \ldots \parallel \alpha_k * \varphi_k x_k \parallel syn(f)) \backslash V$$

where the φ_i 's are applied morphisms and $syn(f)$ controls the correct application of the semantic rules for f.

Let us give some examples of definable operators. Let $w \in \mathbb{M}$ and U be a decidable (i.e. computable as well as its complement) subset of $\mathbb{M}$

(i) discriminated ticking :

$$p \overset{u}{\rightarrow} p', u \notin U \vdash w *_U p \overset{u}{\rightarrow} w *_U p'$$

$$p \overset{u}{\rightarrow} p', u \in U \vdash w *_U p \overset{w.u}{\rightarrow} w *_U p'$$

(ii) discriminated triggering :

$$p \xrightarrow{u} p', u \notin U \vdash w =>_U p \xrightarrow{u} w =>_U p'$$

$$p \xrightarrow{u} p', u \in U \vdash w =>_U p \xrightarrow{w.u} p'$$

(iii) discriminated sum :

$$p \xrightarrow{u} p', u \in U \vdash p \oplus_U q \xrightarrow{u} p'$$

$$q \xrightarrow{v} q', v \in U \vdash p \oplus_U q \xrightarrow{u} q'$$

and the rules of $(p \parallel q)$ with the additional hypotheses $u \notin U, v \notin U$:

$$p \xrightarrow{u} p', u \notin U \vdash p \oplus_U q \xrightarrow{u} p' \oplus_U q$$

$$p \xrightarrow{u} p', q \xrightarrow{v} q', u \notin U, v \notin U \vdash p \oplus_U q \xrightarrow{uv} p' \oplus_U q'$$

$$q \xrightarrow{v} q', u \notin U \vdash p \oplus_U q \xrightarrow{v} p \oplus_U q'$$

(iv) discriminated synchrony :

$$p \xrightarrow{u} p', q \xrightarrow{v} q', u \in U, v \in U \vdash p \oplus_U q \xrightarrow{uv} p' \oplus_U q'$$

and the rules of interleaving when $u \notin U$ or $v \notin U$.

(v) conjunction :

$$p \xrightarrow{u} p', q \xrightarrow{u} q', u \in U \vdash p \cap_U q \xrightarrow{u} p' \cap_U q'$$

and the rules of $(p \parallel q)$ when $u \notin U, v \notin U$

8. Subcalculi : rational parallel place machines

Owing to its universality the MEIJE calculus may appear too strong in some respects (undecidability problems). Therefore it is natural to look for less powerful calculi. We have already seen such a subcalculus, namely that of *finite processes* (par. 6 and [23]) in which one can only define finite transition systems. Many synchronization problems may be formulated in this calculus which is closed under many interesting operators (see [1,24,26]). But it disallows expressing dynamical creation of parallelism. A typical example of "unboundedly parallel" object is that of **bag** (cf. example 4.5 in § 4).

In this section we propose a generalization of finite processes following the usual mathematical step in which "finite" is generalized by "rational".

We have not emphasized algebraic properties of operators (see [22,2]) and their proofs in this notes. At least we may note that

$$\begin{cases} p+q \simeq q+p \\ p+(q+r) \simeq (p+q)+r \\ p+\mathbb{0} \simeq p \end{cases}$$

$$\begin{cases} p \times (q \times r) \simeq (p \times q) \times r \\ (p+q) \times r \simeq p \times r + q \times r \end{cases}$$

and also

$p \times q \simeq q \times p$

$p \times \mathbb{1} \simeq p \ , \ p \times \mathbb{0} \simeq \mathbb{0}$

and moreover

$$\begin{cases} (u+v)\Rightarrow p \simeq u\Rightarrow p \ + \ v\Rightarrow p \\ u\Rightarrow(p+q) \simeq u\Rightarrow p \ + \ u\Rightarrow q \\ 1\Rightarrow p \simeq p \ , \ 0\Rightarrow p \simeq \mathbb{0} \\ u\Rightarrow(v\Rightarrow p) \simeq (u.v)\Rightarrow p \\ u\Rightarrow(p\times q) \simeq (u\Rightarrow p)\times q \end{cases}$$

Therefore the set $\mathcal{A}_{\mathbb{M}}/\sim$ of MEIJE processes is a $\mathbb{K}(\mathbb{M})$- algebra (see [10], chap. VII). Introducing a "star" or **cross** operator

$$\boxed{x^{\times} = (y \text{ where } y = x \times y + \mathbb{1})}$$

we may call, following Eilenberg (idem), the operators

$$p+q \ , \ p \times q \ , \ u\Rightarrow p \ , \ p^{\times}$$

the **rational** ones over $\mathcal{A}_{\mathbb{M}}/\sim$.

Note that, using the notations of SCCS [22]

$$p^{\times} \simeq \sum_{n \in \mathbb{N}} p^n \quad \text{where } p^0 = \mathbb{1}, p^{n+1} = p \times p^n$$

and that the usual identity $p^{\times} \simeq p \times p^{\times} + \mathbb{1}$ is valid.

Reminder : we recall that the family $Rat(\mathbb{M})$ of rational subsets of a monoid $\mathbb{M}$ is (cf. [9,10]) the closure of the family of finite subsets of $\mathbb{M}$ by the rational operations

- union, denoted $U+V$
- product $U.V = \{u.v / u \in U, v \in V\}$
- generated submonoid denoted $U^{\times}$ in order to avoid confusion with the set of sequences.

When $\mathbb{M}$ is a product monoid (for instance $\mathbb{N}^k$, with operation $(n_1,\ldots,n_k)\bullet(m_1,\ldots,m_k) = (n_1+m_1,\ldots,n_k+m_k)$) the rational subsets are usually called rational relations.

We might call *rational* the agents of the family $\mathcal{R}_{\mathbb{M}}$ closure of the calculus $\mathcal{F}_{\mathbb{M}}$ of finite processes by rational operators, but we shall not keep exactly this denomination.

Example 8.1 : in $\mathcal{R}_{\mathbb{M}}$ lie the **rational triggers** τ_U for U a rational subset of $\mathbb{M}$, since τ is a morphism :

$\tau_0 = \mathbb{0}$, $\tau_1 = \mathbb{1}$

(we recall that 0 stands for the empty set, and that a singleton $\{u\}$ is denoted u)

$\tau_{U+V} \simeq \tau_U + \tau_V$

$\tau_{U.V} \simeq \tau_U \times \tau_V$, $\tau_{u.V} \simeq u\Rightarrow t_V$

$\tau_{U^{\times}} \simeq (\tau_U)^{\times}$

Let us try to see what the transition systems determined by agents in $\mathcal{R}_{\mathbb{M}}$ are. We start with finite processes, expressed by agent of $\mathcal{F}_{\mathbb{M}}$ in normal form :

$$(x_l \text{ where } ..., x_i = \sum_{1 \le j \le k} u^j{}_i : x_j, ...)$$

The operators $p+q, p \times q$ and $u => p$ preserve finiteness, thus let us look at the effect of the cross operator. It may be (recursively) specified by

$$\begin{cases} \vdash p^{\times} \xrightarrow{1} \mathbb{1} \\ p \xrightarrow{u} q \ , p^{\times} \xrightarrow{v} r \vdash p^{\times} \xrightarrow{u\,v} q \times r \end{cases}$$

or more explicitely :

$$p \xrightarrow{u_1} p_1, ..., p \xrightarrow{u_n} p_n \vdash p^{\times} \xrightarrow{u_1 ... u_n} p_1 \times ... \times p_n$$

(including the case $n=0$ where $p^{\times} \xrightarrow{1} \mathbb{1}$).

Thus starting with $p \in \mathcal{F}_{\mathbb{M}}$ the states of the transition system determined by $p^{\times}$ have the form

$$((x^{n_1}{}_1 \times ... \times x^{n_k}{}_k \ \textbf{where} \ ..., x_i = \sum_{1 \le j \le k} u^j{}_i : x_j, ...)$$

(excepted for $p^{\times}$ itself)

This means that given the finite transition table, states may be regarded as belonging to $\mathbb{N}^k$. For example (8.2) if

$$p = (x \text{ where } x = a : y, y = b : x + c : y)$$

$$q = (y \text{ where } x = a : y, y = b : x + c : y)$$

the states reachable from $p^{\times}$ are the $p^n \times q^m$ and the transitions are given by

$$p^n \times q^m \simeq \sum_{k \le m} a^n b^{m-k} c^k : (p^{m-k} \times q^{n+k})$$

This suggests the following definition :

a **simple parallel place machine** on the monoid $\mathbb{M}$ of actions is a structure $\mathcal{A} = (\mathbb{P}, \mathbb{M}, T, I)$ where

- $\mathbb{P} = \{p_1, ..., p_k\}$ is a finite set of **places**
- $T \subseteq \mathbb{P} \times \mathbb{M} \times \mathbb{P}$ is a finite set of **elementary transitions**
- I is the set of **initial markings** which is a rational subset of $\mathbb{N}^{\mathbb{P}}$.

Intuitively each place contains **tokens** and a marking μ sets tokens into place, i.e. is a mapping of $\mathbb{N}^{\mathbb{P}}$ which itself is identified to $\mathbb{N}^k$. The **states** of the machine are markings. Each elementary transition (p_i, u, p_j) removes one token from p_i and, performing u , put it in p_j. Then the behaviour is the following : given a marking μ , the machine performs synchronously elementary transitions. It thus performs a product of such transitions, by which all tokens are moved. Formally :

- each place p_i is identified to the marking where there is one token in p_i and no other one.
- T is identified to a subset of the product monoid $\mathbb{N}^{\mathbb{P}} \times \mathbb{M} \times \mathbb{N}^{\mathbb{P}}$.

For instance, the elementary transitions in the above example are :

$$(1,0) \xrightarrow{a} (0,1) \ , \ (0,1) \xrightarrow{b} (1,0) \text{ and } (0,1) \xrightarrow{c} (0,1).$$

Then the set of **transitions** of the machine $\mathcal{A}$ is $T^\times$, the submonoid of $\mathbb{N}^{\mathbb{P}} \times \mathbb{M} \times \mathbb{N}^{\mathbb{P}}$ generated by T.

Example 8.2 (continued) :

$$T^\times = \{(n,m) \xrightarrow{a^n b^{m-k} c^k} (m-k,n+k) / n,m \in \mathbb{N}, k \leq m\}$$

In order to take into account the set of initial states, we define the transition system determined by a simple place machine $\mathcal{A} = (\mathbb{P},\mathbb{M},T,I)$ by

$\Xi(\mathcal{A}) = (\mathbb{Q},\mathbb{M},U)$ where

- $\mathbb{Q} = \mathbb{N}^{\mathbb{P}} \cup \{I\}$
- $(q,u,q') \in U \iff q = I \;\&\; \exists q'' \in I.\ (q'',u,q') \in T^\times$ or $(q,u,q') \in T^\times$

This definition yields the

Proposition 8.1

(1) For any simple parallel place machine $\mathcal{A}$ on $\mathbb{M}$, $\Xi(\mathcal{A})$ is strongly realizable in $\mathcal{R}_{\mathbb{M}}$

(2) any agent $p \in \mathcal{R}_{\mathbb{M}}$ strongly realizes $\Xi(\mathcal{A})$ for some simple parallel place machine $\mathcal{A}$.

From a semantical point of view, simple parallel place machines only correspond to a special kind of what might been called *rational transition systems*. These are $\Theta = (\mathbb{Q},\mathbb{M},T)$ *where*

- $\mathbb{Q}$ is a commutative monoid of states
- $\mathbb{M}$ is a commutative monoid of actions
- $T \subseteq \mathbb{Q} \times \mathbb{M} \times \mathbb{Q}$ is a rational relation

However these are rather abstract objects, thus we focus on the case where $\mathbb{Q}$ is the set of distribution of tokens in places. Therefore a **rational parallel place machine** (on $\mathbb{M}$) is

$\mathcal{A} = (\mathbb{P},\mathbb{M},T,i)$ where

- $\mathbb{P} = \{p_1,\ldots,p_k\}$ is a finite set of places
- $\mathbb{M}$ is a commutative monoid of actions
- $T \subseteq \mathbb{N}^{\mathbb{P}} \times \mathbb{M} \times \mathbb{N}^{\mathbb{P}}$ is a rational relation
- $i \in \mathbb{N}^{\mathbb{P}}$ is the initial state

As before, this machine determines a transition system with initial state $\Xi(\mathcal{A}) = (\mathbb{N}^{\mathbb{P}},\mathbb{M},T,i)$. It corresponds to a synchronous behaviour : all the tokens must be moved in a transition.

Example : bags are objects of this kind, determined by a one place machine. For if we denote :

$\varepsilon = (1) \xrightarrow{1} (1)$ (a "neutral" transition)

$t_1 = (0) \xrightarrow{a} (1)$ (to put a token)

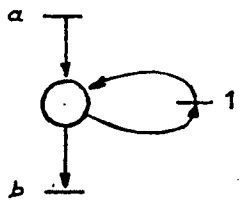
$t_2 = (1) \xrightarrow{b} (0)$ (to remove a token)

the transition relation $T = \varepsilon^\times \bullet (t_1 + t_2) \bullet (t_1 + t_2)^\times$ is such that

$$T = \{(k) \xrightarrow{a^n b^m} (k+n-m) / n+m>0, m \leq k\}$$

(cf. example 4.5)

The transitions ε, t_1, t_2 can be figured, with labelled barred lines to make the distinction with the transition system itself, by



Obviously each **rational clock** i.e. h_U with U a rational subset of $\mathbb{M}$ is also defined by a rational machine (with one place and no tokens).

If we want to get an *asynchronous behaviour* we associate with $\mathcal{A}$ the transition system

$$\Delta(\mathcal{A}) = (\mathbb{N}^{\mathbb{P}}, \mathbb{M}, T', i)$$

where

$$T' = \{\varepsilon_1 + \ldots + \varepsilon_k\}^{\times} \bullet T$$

with

$$\varepsilon_j = (\mu_j, 1, \mu_j) \quad \mu_j(p) = \begin{cases} 1 & \text{if } p = p_j \\ 0 & \text{otherwise} \end{cases}$$

(elementary "neutral" transitions)

In the asynchronous system $\Delta(\mathcal{A})$ a transition of the machine may be performed as soon as there are enough tokens on the places since

$$(\mu_1, u, \mu_2) \in T' \iff \exists \mu'_1, \mu'_2 \bullet (\mu'_1, u, \mu'_2) \in T \;\&$$
$$\mu'_1 \leq \mu_1 \;\&\; \mu_2 = \mu_1 - \mu'_1 + \mu_2$$

For instance the bag is an example of asynchronous system.

Note : The sets $\{\mu' \;/\; \exists u . \mu \overset{u}{\rightarrow} \mu'\}$ and $\{u \;/\; \exists \mu' . \mu \overset{u}{\rightarrow} \mu'\}$ for $\mu \in \mathbb{N}^{\mathbb{P}}$ are rational (got from T by transductions).

We could have allowed a rational set I of initial states and devised $\Xi(\mathcal{A})$ as in the simple case ; but nothing is gained in doing so :

Lemma 8.1

Let $\mathcal{A} = (\mathbb{P}, \mathbb{M}, T, I)$ be a generalized rational place machine, i.e. $I \in Rat(\mathbb{N}^{\mathbb{P}})$. Then there exists a rational parallel place machine $\mathcal{A}' = (\mathbb{P}', \mathbb{M}, T', i)$ such that $\Xi(\mathcal{A})$ and $\Xi(\mathcal{A}')$, restricted to states reachable from the initial ones, are isomorphic.

In the proof one uses some results of Eilenberg & Schutzenberger [9]. We only mention the fact that *non-conservative* transitions, which lose some tokens, are introduced.

We have noted that rational clocks are determined by some rational parallel place machines. In order to define these clocks in MEIJE we have to introduce a new (temporal) **iterator**. Once remarked that

$$h_U \simeq \tau_U \times (1 : h_U)$$

we let

$$\boxed{x^{\alpha} = (y \text{ where } y = x \times (1 : y))}$$

The specification of this operator is

$$p \overset{u}{\rightarrow} q \vdash p^{\square} \overset{u}{\rightarrow} q \times p^{\square}$$

For the rest of this section $\mathbb{M} = \mathbb{N}\langle A \rangle$ and $A = \{a_1,...,a_m\}$.

Let $\mathcal{H}_{\mathbb{M}}$ be the closure of $\mathcal{R}_{\mathbb{M}}$ by the family of operators

- $u \Rightarrow p$ for $u \in \mathbb{M}$
- φp for $\varphi = \langle u_1/a_1,...,u_m/a_m \rangle, u_i \in \mathbb{M}$
- $p \cap_{B^*} q$ (see § 7) which is denoted $p \wedge_B q$, for $B \subseteq A$
- $p^{\square}$

For instance $h_U \in \mathcal{H}_{\mathbb{M}}$ for any $U \in Rat(\mathbb{M})$ since $h_U \simeq \tau_U{}^{\square}$.

Proposition 8.2

(1) for any rational place machine $\mathcal{A}$ on $\mathbb{M} = \mathbb{N}\langle A \rangle$, $\Xi(\mathcal{A})$ is strongly realizable in $\mathcal{H}_{\mathbb{N}\langle A' \rangle}$ for some A' $(A \subseteq A')$

(2) any agent $p \in \mathcal{H}_{\mathbb{M}}$ strongly realizes $\Xi(\mathcal{A})$ for some rational place machine on $\mathbb{M}$.

For the first point, the argument is similar to that of Theorem 7.1. Let $\mathcal{A} = (\mathbb{P}, \mathbb{M}, T, i)$ be a rational machine with $\mathbb{P} = \{p_1,...,p_k\}$, $\mathbb{M} = \mathbb{N}\langle a_1,...,a_m \rangle$.

Let $B = \{b_1,...,b_k, c_1,...,c_k\}$ be a set of new action symbols (not in A). Obviously $T \subseteq \mathbb{N}^{\mathbb{P}} \times \mathbb{M} \times \mathbb{N}^{\mathbb{P}}$ is isomorphic to a rational subset U of $\mathbb{N}\langle A \cup B \rangle$. We interpret b_i as the presence of an "input" token on the place p_i and similarly c_i as the presence of an "output" token. We have to control the flow of tokens in the clock h_U "realizing" the transitions. We need a synchronizer t such that

$$t \xrightarrow{c^{n_1}_1 ... c^{n_k}_k} ((b^{n_1}_1 ... b^{n_k}_k) \Rightarrow t)$$

for all $(n_1,...,n_k) \in \mathbb{N}^{\mathbb{P}}$

(in fact $(b^{n_1}_1 ... b^{n_k}_k) \Rightarrow t$ is a "synchronous bag" containing $n_1,...,n_k$ coloured tokens).

This achieved if we let

$$t = \Big(\sum_{1 \le i \le k} c_i : b_i : 1\!\!1 \Big)^{\times \square}$$

(which is in $\mathcal{H}_{\mathbb{N}\langle A \cup B \rangle}$)

Let φ_B be the projection $\langle 1/b_1,...,1/b_k, 1/c_1,...,1/c_k \rangle$.

Then with each state $\mu = (l_1,...,l_k)$ in $\mathbb{N}^{\mathbb{P}}$ we associate

$$\vartheta(\mu) = \varphi_B(h_U \wedge_B ((b^{l_1}_1,...,b^{l_k}_k) \Rightarrow t))$$

and this is the claimed strong realization.

For the second point one uses Eilenberg & Schutzenberger ' s results [9] again and introduce *non-conservative* transitions which this time create tokens.

The proposed elements in this section do not allow to appreciate the practical and theoretical interest of rational parallel place machines. Moreover a similar study might be attempted for "rationally specified" operators. At least we may note that rational place machines generalize Petri nets -which we showed to be strongly realizable in MEIJE, in a more direct way, cf. [3]. Let us briefly face this. A **Petri net** (with which we assume the reader familiar, see for instance [27]) is a structure $(\mathbb{P}, A, Pre, Post)$ where

- $\mathbb{P} = \{p_1,...,p_k\}$ is the finite set of places
- $A = \{a_1,...,a_m\}$ is the finite set of transitions
- $Pre : \mathbb{P}\times A \to \mathbb{N}$ and $Post : A\times\mathbb{P} \to \mathbb{N}$ are the numerical functions setting the preconditions and postconditions to the firing of transitions.

A transition $a\in A$ is *enabled* by the marking $\mu\in\mathbb{N}^{\mathbb{P}}$ if for all place $p\in\mathbb{P}$ $\mu(p)\geq Pre(p,a)$. And by firing a we get the marking μ' :

$$\mu'(p) = \mu(p) - Pre(p,a) + Post(a,p)$$

Let $t_i = (\mu_i, a_i, \mu'_i)$ for $1\leq i\leq m$ where $\mu_i(p) = Pre(p,a_i)$ and $\mu'_i = Post(a_i,p)$
Then the behaviour of the Petri net with initial marking μ is exactly that of the asynchronous transition system $\Delta(\mathcal{A})$ (which obviously is also $\Xi(\mathcal{A}')$ for some $\mathcal{A}'$) associated with the machine

$$\mathcal{A} = (\mathbb{P}, \mathbb{N}<A>, (t_1+...+t_m), \mu)$$

Remark : it is well-known (see [9]) that for any rational transition relation T there exists a *finite* set $U \subseteq \mathbb{N}^{\mathbb{P}}\times\mathbb{M}\times\mathbb{N}^{\mathbb{P}}$ of transitions such that T is a rational subset of U^*. Such a finite set U may be represented as a labelled Petri net. Thus a rational parallel place machine may also be regarded as a (rationally) synchronized Petri net : a labelled Petri net provided with a rational expression on its transitions.

9. Conclusion

We chose to interpret our language in the semantical universe of transition systems. These are rather general and conceptually simple mathematical objects. As demonstrated by Plotkin [28] transition systems provide a general framework to express semantics. They also set up the natural universe in which to interpret formulae of some modal logics. For all these reasons the notion of transition system is a basis for comparison of various formalisms (see for example [4,5,18]. In these notes they have been mainly used to evaluate expressiveness of our MEIJE calculus - something which could not have been achieved without the very fruitful idea of monoid of actions due to Milner [21,22].

Obviously this is not the only possible approach to the question : what is a process ? A great variety of models may be found in the literature, among which we would like (unfairly) distinguish :

(1) models which lay "below" transition systems. One of the best known examples is that of formal languages. Although they are sometimes taken as a direct interpretation of some syntax or algebra (as in [16,17,25,30,32]), formal languages have been for a long time strongly linked to transition systems (or automata, see [10]). This is also the case in the area of parallelism (cf. [1,24,26]) where they are very often assorted by some relevant informations (see [6,7,8]). Indeed these models seem to correspond to some canonical kinds of transition systems. These models own the advantage to provide a direct semantics (without recourse to quotienting), with a direct definition of the operators ; this has not been achived here.

In the same category one may also put the observational semantics of Hennessy [14,15] which is near ours since it lies upon operational semantics. As Darondeau [8] points out this semantics contrasts with bisimulations in that backtracking along the behaviour of a process is not allowed. It seems also to be a question of linear vs branching time.

It is not very surprising that, among all the existing definitions of semantic equality, bisimulations are very strong as shown by Brookes and Rounds in [5]. However in some respects our equipollences determined by observation criteria on actions may appear to be too weak. To see why, let us say a few words about the second approach.

(2) On the other side one find models of the notion of processes which are "above" transition systems. The concept of machines or automata belongs here. They naturally determine transitions between states. In our domain this category is mainly represented by Petri nets (see [27]). The general idea is : how do things operate ?

In machines, the structure of states may carry some meaningful information (e.g. : boundedness in Petri nets) ; this structure requires a fine description. Therefore if one wants to show for instance that a calculus or language of processes corresponds to some class of machines (cf. for instance [3,23]) one has to carefully examine what properties are used in the proof of that correspondance. From that may emerge a complete axiomatization of the calculus ; we note that completeness is usually shown by means of normal forms, in which one recognizes the structure of a machine ([23]) or more generally of a concrete interpretation ([12,14]).

We have not paid much attention to proof theory in these notes. Many algebraic properties are valid in our calculi (cf. [2,22]) and the proof principle, i.e. Park's induction for congruences provides a very powerful tool. However we would like to have a more syntactical version, explicitely involving recourse to specifications of operators (a similar aim holds for proofs of simulation) ; this is a research in progress. More generally it should be clear that we have emphasized here an operational or "syntactical" point of view. Let us quote Milner about this subject : "operational semantics, since it can be set up with so few preconditions, must be the touchstone for assessing mathematical models rather than the reverse" ([22]). Not less clearly, much work remains to be done towards a smooth theory of that matter.

Acknoledgement : many ideas and results presented in these notes were elaborated with the contribution of R. de Simone, and some are his own.

REFERENCES

1 A. Arnold & M. Nivat : "Comportements de processus", Coll. AFCET "Les Mathématiques de l'Informatique" (AFCET, Paris, 1981)

2 D. Austry & G. Boudol : "Algèbre de processus et synchronisations", Theoret. Comput. Sci. 30 (1984) 91-131

3 G. Boudol, G. Roucairol & R. de Simone : "Petri nets and algebraic calculi of processes", Rapport de Recherche INRIA 292 (1984)

4 S.D. Brookes : "On the relationship of CCS and CSP", ICALP 83, Lecture Notes in Comput. Sci. 154 (1983) 83-96

5 S.D. Brookes & W.C. Rounds : "Behavioural equivalence relations induced by programming logics", ICALP 83, Lecture Notes in Comput. Sci. 154 (1983) 97-108

6 S.D. Brookes, C.A.R. Hoare & A.W. Roscoe : "A theory of communicating sequential processes", JACM 31 (1984) 560-599

7 Ph. Darondeau & L. Kott : "On the observational semantics of fair parallelism", ICALP 83, Lecture Notes in Comput. Sci. 154 (1983) 147-159
s.a. Rapport de Recherche INRIA 262 (1983)

8 Ph. Darondeau : "Infinitary languages and fully abstract models of fair asynchrony", Advanced Course on Logics and Models for Verification and Specification of Concurrent Systems, La Colle-sur-Loup (1984)

9 S. Eilenberg & M.P. Schutzenberger : "Rational sets in commutative monoids", Journal of Algebra 13 (1969) 173-191

10 S. Eilenberg : "Automata, Languages and Machines", vol.A, Academic Press (1974)

11 J.S. Gourlay, W.C. Rounds & R. Statman : "On properties preserved by contractions of concurrent systems", Intern. Symp. on Semantics of Concurrent Computations, Evian 79, G. Kahn Ed., Lecture Notes in Comput. Sci. 70 (1979) 51-65

12 M. Hennessy & R. Milner : "On observing nondeterminism and concurrency", ICALP 80, Lecture Notes in Comput. Sci. 85 (1980) 299-309
s.a. "Algebraic laws for nondeterminism and concurrency" CSR-133-83, Computer Science Dept., Edinburgh Univ. (1983)

13 M. Hennessy, Wei Li & G. Plotkin : "A first attempt at translating CSP into CCS", 2 nd Intern. Conf. on Distributed Computing Systems, Paris (1981) 105-115

14 M. Hennessy & R. de Nicola : "Testing equivalences for processes", ICALP 83, Lecture Notes in Comput. Sci. 154 (1983) 548-560
s.a. CSR-123-82, Comput. Sci. Dept., Edinburgh Univ. (1982)

15 M. Hennessy : "Modelling finite delay operators", CSR-153-83, Comput. Sci. Dept., Edinburgh Univ. (1983)

16 C.A.R. Hoare : "A model for communicating sequential processes", Techn. Rep. PRG-22, Programming Research Group, Oxford Univ. (1981)

17 M. Jantzen : "The power of synchronizing operations on strings", Theoret. Comput. Sci. 14 (1981) 127-154

18 K. Jensen : "A method to compare the descriptive power of different types of Petri nets", MFCS 80, Lecture Notes in Comput. Sci. 88 (1980) 348-361

19 R.M. Keller : "Formal verification of parallel programs", CACM 19 (1976) 371-384

20 R. Milner : "A Calculus of Communicating Systems", Lecture Notes in Comput. Sci 92 (1980)

21 R. Milner : "On relating synchrony and asynchrony", CSR-75-80, Comput. Sci. Dept., Edinburgh Univ. (1981)

22 R. Milner : "Calculi for synchrony and asynchrony", Theoret. Comput. Sci. 25 (1983) 267-310

23 R. Milner : "A complete inference system for a class of regular behaviour", J. of Computer and Systems Sciences 28 (1984) 439-466

24 M. Nivat : "Synchronization of concurrent processes", in "Formal Language Theory : Perspective and Open Problems", R.V. Book, ed., A.P. (1980) 429-454

25 W.F. Ogden, W.E. Riddle & W.C. Rands : "Complexity of expressions allowing concurrency", 5^{th} ACM POPL (1978) 185-194

26 D. Park : "Concurrency and automata on infinite sequences", 5^{th} GI Conf., Lecture Notes in Comput. Sci. 104 (1981) 167-183

27 J.L. Peterson : "Petri Nets", ACM Computing Surveys 9 (1977) 223-252

28 G. Plotkin : "A structural approach to operational semantics", Daimi FN-19, Comput. Sci. Dept., Aarhus Univ. (1981)

29 G. Plotkin : "An operational semantics for CSP", in "Formal Description of Programming Concepts II", D. Bjorner, ed, North- Holland (1982) 199-225

30 A.C. Shaw : "Software description with flow expressions", IEEE Trans. on Software Engineering 4 (1978) 242-254

31 J. Sifakis : "Property preserving homomorphisms of transition systems", Logics of Programs, Lecture Notes in Comput. Sci. 164 (1984) 458-473

32 R. de Simone : "Langages infinitaires et produit de mixage", Theoret. Comput. Sci. 31 (1984) 83-100

33 R. de Simone : "Calculabilité et expressivité dans l'algèbre de processus parallèles Meije", Thèse de 3e cycle, Univ. Paris 7 (1984)

34 R. de Simone : "Note on Meije and SCCS : infinite sum operators vs non-guarded definitions", Theoret. Comput. Sci. 30 (1984) 133-138

35 G. Winskel : "Synchronization trees", CMU-CS-83-139, Comput. Sci. Dept., Carnegie-Mellon Univ. (1983)
s.a. ICALP 83, Lecture Notes in Comput. Sci. 154 (1983) 695-711

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique